

2021 秋《计算机硬件基础》

02_1 数制与运算 作业

作业一 概述

一、 填空题

- 1、将二进制、八进制和十六进制数转换为十进制数的共同规则是 乘权求和。
- 2、二进制数 $(1101011.011)_B$ 对应的八进制数为(153.3)_O，对应的十六进制数为(6B.6)_H。
- 3、 $(100101010011.00110111)_{8421BCD}$ 表示的十进制数为(953.37)_D。
- 4、 $(001111110001.01011111)_{2421BCD}$ 表示的十进制数为(391.59)_D。
- 5、 $(100101111100.00010100)_{5211BCD}$ 表示的十进制数为(647.12)_D。
- 6、 $(110010100111.10000111)_{\text{余}3BCD}$ 表示的十进制数为(974.54)_D。
- 7、数字字符“9”对应的 ASCII 码为 57。
- 8、表示任意一个 3 位十进制数至少需要 10 位二进制数。

二、 选择题

- 1、十进制数 127.25 对应二进制数为 (A)。
A、1111111.01 B、10000000.10
C、1111110.01 D、1100011.11
- 2、十进制数 28.43 的余 3BCD 码是 (B)。
A、00111000.01000011 B、01011011.01110110
C、01101100.10000111 D、01111101.10011000
- 3、在下列 ASCII 字符中，最大的字符是 (B)。
A、“A” B、“z”
C、“9” D、“0”
- 4、采用 CRC 进行差错校验，生成多项式为 $G(X)=X^4+X+1$ ，信息码字为 10111，则计算出的 CRC 校验码是 (D)。
A. 0000 B. 0100 C. 0010 D. 1100

三、 计算题

- 1、将十进制数 33.628 转换成等值的二进制数和十六进制数，要求写出转换过程，且二进制数保留小数点以后 4 位有效位。

转换为二进制：100001.1010

$$\begin{array}{r} 2 \overline{) 33} \\ \underline{26} \\ 8 \\ \underline{6} \\ 2 \\ \underline{2} \\ 0 \end{array} \quad \begin{array}{l} \text{余数 } 1 \\ \text{余数 } 0 \\ \text{余数 } 0 \\ \text{余数 } 0 \\ \text{余数 } 0 \\ \text{余数 } 1 \end{array}$$

整数部分 100001

转换为二进制 100001.1010

$$\begin{array}{r} .628 \\ \times 2 \\ \hline 1.256 \\ \times 2 \\ \hline 2.512 \\ \times 2 \\ \hline 5.024 \\ \times 2 \\ \hline 10.048 \end{array} \quad \begin{array}{l} \text{进位 } 1 \\ \text{进位 } 0 \\ \text{进位 } 1 \\ \text{进位 } 0 \end{array}$$

小数部分

转换为十六进制：21.A （也可由前述二进制转换而来）

$$\begin{array}{r}
 16 \overline{) 33} \\
 \underline{16} \\
 17 \\
 \underline{16} \\
 1
 \end{array}
 \begin{array}{l}
 \text{余数1} \uparrow \\
 \text{余数2} \uparrow
 \end{array}
 \begin{array}{r}
 . 628 \\
 \times 16 \\
 \hline
 10.048
 \end{array}
 \begin{array}{l}
 \text{进位A} \\
 \text{进位A}
 \end{array}$$

整数部分 小数部分

转换为十六进制 21.A

2、将二进制数 11.01101 转换成等值的十进制数和十六进制数，要求写出转换过程。

转换为十进制：3.40625

$$\begin{aligned}
 & (11.01101)_2 \\
 &= 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} + 0 \times 2^{-4} + 1 \times 2^{-5} \\
 &= 2 + 1 + 0 + 0.25 + 0.125 + 0 + 0.03125 \\
 &= 3.40625
 \end{aligned}$$

转换为十六进制：3.68

$$\begin{aligned}
 11.01101 &= (0011.01101000)_2 = (3.68)_{16} \\
 & \quad (3.68)_{16}
 \end{aligned}$$

3、写出下列数的原码、反码、补码。

	+1001	-1001	+0.1001	-0.1001	+0000	-0000
原码	01001	11001	01001	11001	00000	10000
反码	01001	10110	01001	10110	00000	11111
补码	01001	10111	01001	10111	00000	00000

4、已知下列机器数，写出相对应的真值，注意写出推导过程。

(1) [X]_原=10111; (2) [X]_反=10111; (3) [X]_补=10111。

(1) [X]_原=10111

符号位：1 表示符号为-

真值的绝对值：(111)₂=4+2+1=7

真值：-7

(2) [X]_反=10111

转为原码：符号位 1 不变，其余四位 0111 各位取反 1000，即 11000

符号位：1 表示符号为-

真值的绝对值: $(1000)_2 = 8$

真值: -8

(3) $[X]_{\text{补}} = 10111$

转为反码: 符号位 1 不变, 其余四位 0111 减 1, 即 10110

转为原码: 符号位 1 不变, 其余四位 0110 各位取反 1001, 即 11001

符号位: 1 表示符号为-

真值的绝对值: $(1001)_2 = 8+1=9$

真值: -9

5、采用二进制补码求 $21+18$ 、 $21-18$ 、 $-21+18$ 、 $-21-18$ 。要求有分析过程, 并列采用十进制数和二进制补码进行运算的各算式。求得的 $(-21+18)_D$ 的补码是多少? 你是怎样验证结果是否正确的?

(1) 计算结果

$21+18$: 39

十进制	原码	反码	补码
21	00010101	00010101	00010101
18	00010010	00010010	00010010

$$\begin{array}{r} 21+18: \quad 00010101 \\ + \quad 00010010 \\ \hline 00100111 \end{array} \quad \text{即 } 39$$

$21-18$:

十进制	原码	反码	补码
21	00010101	00010101	00010101
-18	10010010	11101101	11101110

$$\begin{array}{r} 21-18: \quad 00010101 \\ + \quad 11101110 \\ \hline 10000011 \end{array} \quad \text{即 } 3$$

$-21+18$:

十进制	原码	反码	补码
-21	10010101	11101010	11101011
18	00010010	00010010	00010010

$$\begin{array}{r} -21+18: \quad 11101011 \\ + \quad 00010010 \\ \hline 11111101 \end{array} \quad \begin{array}{l} \text{补码 } 11111101 \\ \text{反码 } 11111100 \\ \text{原码 } 10000011 \end{array} \quad \text{即 } -3$$

$-21-18$:

十进制	原码	反码	补码
-21	10010101	11101010	11101011
-18	10010010	11101101	11101110

$$\begin{array}{r} -21-18: \quad 11101011 \\ + \quad 11101110 \\ \hline 11011001 \end{array} \quad \begin{array}{l} \text{补码: } 11011001 \\ \text{反码: } 11011000 \\ \text{原码: } 10100111 \end{array} \quad \text{即 } -39$$

(2) 补码验证

$(-21+18)_D$ 的补码: 11111101

验证: 求补码的补码应当得到原码

11111101 求反码得: 10000010 再加一得: 10000011

6、写出以下两个 IEEE 754 单精度浮点数对应的十进制数。

(1) 44900000H (2) C6801600H

(1) 44900000H

二进制: 01000100100100000000000000000000

数符 S: 0

阶码 E: 10001001

尾数 M: 001000000000000000000000

10001001 即 2^{10} 即 $(10000000000)_2$

001000000000000000000000 即 $(1.001)_2$

故表示 $+(1.001 \times 10000000000)_2$, 即 1152

(2) C6801600H

二进制: 11000110100000000001011000000000

数符 S: 1

阶码 E: 10001101

尾数 M: 000000000010110000000000

10001101 即 2^{14} 即 $(1000000000000000)_2$

000000000010110000000000 即 $(1.00000000001011)_2$

故表示 $-(1.00000000001011 \times 1000000000000000)_2$ 即 -16935

7、设两浮点数 $x=2^{(+010)} \times 0.110100$, $y=2^{(+100)} \times (-0.101010)$, 求 $x+y$ 。

$x=2^{(+010)} \times 0.110100 = 2^{(+100)} \times 0.001101$ 尾数 0001101

$y=2^{(+100)} \times (-0.101010)$ 尾数 1010110

尾数相加 1100011 即 -0.011101

$x+y=2^{(+100)} \times (-0.011101) = 2^{(+011)} \times (-0.111010)$

十进制: -1856

8、设两浮点数 $x=2^{(-001)} \times (-0.100010)$, $y=2^{(-100)} \times (0.010110)$, 用浮点运算方法计算 $x \times y$, 运算结果的尾数取单字长 (含符号位共 7 位), 舍入规则用 “0 舍 1 入” 法。

阶码相加: $-001+-100=-101$

尾数相乘: $-0.100010 \times 0.010110 = -0.0010111011 = -0.001100$

$x \times y = 2^{(-101)} \times (-0.001100)$

四、 问答题

1、格雷码与 8421 码相比有什么优点? 8421 码有什么缺点?

格雷码与 8421 码相比有什么优点?

格雷码任意两个相邻的代码只有一位二进制数不同, 另外由于最大数与最小数之间也仅一位数不同, 即 “首尾相连”。在数字系统中, 常要求代码按一定顺序变化, 若采用 8421 码, 则数 0111 变到 1000 时四位均要变化, 而在实际电路中, 4 位的变化不可能绝对同时发生, 则计数中可能出现短暂的其它代码 (1100、1111 等)。在特定情况下可能导致电路状态错误或输入错误。使用格雷码可以避免这种错误。

8421 码有什么缺点?

① 如上述, 8421 码相邻两数字之间变化时, 需要改变的位数多

- ② 8421 码 4 位数字表达的数字超过了 0-9 的范畴,在运算时可能会产生不合法的数字,如 1111 (实际上表示了 15)

2、为什么绝大多数计算机中采用补码而不是采用原码或者反码表示机器数?

- ① 补码中 0 的表示唯一,保证了系统编码的连续性和一致性,同时避免了+/- 0 的窘境。
- ② 机器实现加、减运算的方法统一,省去计算机判断符号位或者说判断+/-运算的麻烦。采用补码表示后,不管是加法还是减法都是加法运算。
- ③ 符号位参加运算,不需要单独处理。

补码的以上特点便于硬件设计,实现逻辑简单,更适用于机器实现。

3、常用的二-十进制编码有哪些?为什么说用 4 位二进制代码对十进制数的 10 个数字信号进行编码的方案有很多?

常用的二-十进制编码有哪些?

常用的二-十进制编码,可分为有权码和无权码两类。

有权码:

8421 码、2421 码、5221 码、5421 码

无权码:

余 3 码、余 3 循环码、格雷码

为什么说用 4 位二进制代码对十进制数的 10 个数字信号进行编码的方案有很多?

4 位二进制代码可以表示的数字信号种类有 2^4 即 16 种,因此对 10 个数字信号编码可以有多种选择方式。

4、二进制正数和负数的原码、反码和补码三者之间是什么关系?

正数:原码和反码、补码相同。

负数:原码除符号位外,原码各位取反,得负数的反码。反码加 1,得负数的补码。

2021 秋《计算机硬件基础》

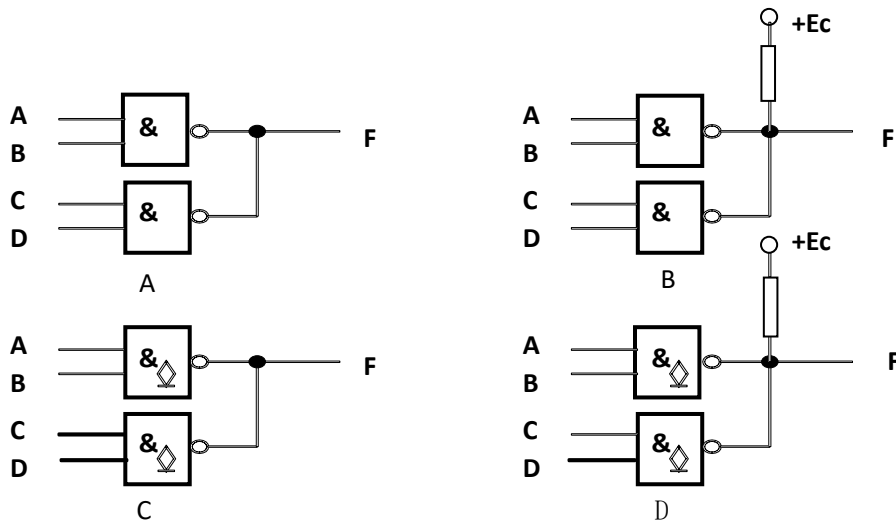
02_2 布尔代数 作业

一、填空题

- 1、逻辑变量和函数只有 0, 1 两种取值，而且它们只是表示两种不同的逻辑状态。
- 2、逻辑函数 $F = \overline{A}\overline{B} + CD$ ，其反函数 $\overline{F} = \underline{(A+B)(\overline{C} + \overline{D})}$ ，其对偶式 $F^* = \underline{(\overline{A} + \overline{B})(C + D)}$ 。
- 3、函数 $F = A\overline{B} + AC + \overline{C}D + ADE$ 的最简与或式是 $A\overline{B} + AC + \overline{C}D$ 。

二、 选择题

- 1、为实现 $F = \overline{AB} \cdot \overline{CD}$ ，下列电路接法正确的是 (D)。



- 2、函数 $F = AB + \overline{A}C + \overline{B}C + \overline{C}D + \overline{D}$ 的最简与或式为 (A)。
A、1 B、0
C、AB D、以上均不是
- 3、逻辑函数 $F = \overline{A + BC(A+B)}$ ，当 ABC 的取值为 (B) 时，F=1。
A、000 B、011
C、101 D、111
- 4、函数 $A \oplus B$ 与 $\overline{A} \oplus \overline{B}$ (C)。
A、互为反函数 B、互为对偶式
C、相等 D、答案都不正确

三、 问答与计算题

说明：分析与计算题要求写出分析推导过程，给出必要的公式。

- 1、将逻辑函数 $F = ABC + \bar{A}BD$ 写成标准与或表达式。

$$\begin{aligned} F &= ABC + \bar{A}BD \\ &= ABC(D + \bar{D}) + \bar{A}BD(C + \bar{C}) \\ &= ABCD + ABC\bar{D} + \bar{A}BCD + \bar{A}B\bar{C}D \end{aligned}$$

- 2、推导出函数 $F = \overline{AB + BC} + A\bar{C}$ 的最简与或式。

$$\begin{aligned} F &= \overline{AB + BC} + A\bar{C} \\ &= \overline{B(A + C)} + A\bar{C} \\ &= \bar{B} + \bar{A}\bar{C} + A\bar{C} \\ &= \bar{B} + \bar{C}(\bar{A} + A) \\ &= \bar{B} + \bar{C} \end{aligned}$$

- 3、列出下述问题的真值表，利用最小项推导法写出其逻辑函数表达式并利用公式简化法进行简化。

有 3 个温度检测器，当检测的温度超过 60℃时，温度检测器输出信号为 1；低于 60℃时，输出为 0。当两个或两个以上的温度检测器的输出为 1 时，总控制器的输出为 1，以控制调控设备，使温度降低到 60℃以下。

真值表

检测器 A	检测器 B	检测器 C	总控制器 F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

逻辑函数表达式

$$F = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

公式简化

$$\begin{aligned} F &= \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC \\ &= (\bar{A}BC + ABC) + (A\bar{B}C + ABC) + (AB\bar{C} + ABC) \\ &= (A + \bar{A})BC + (B + \bar{B})AC + (C + \bar{C})AB \\ &= AB + BC + AC \end{aligned}$$

- 4、用公式法证明下列等式：

$$\bar{B}\bar{C}\bar{D} + \bar{B}CD + ACD + \bar{A}\bar{B}CD + \bar{A}BCD + B\bar{C}\bar{D} + BCD = B + C$$

$$\begin{aligned}
& \bar{B}C\bar{D} + B\bar{C}D + ACD + \bar{A}B\bar{C}\bar{D} + \bar{A}\bar{B}CD + B\bar{C}\bar{D} + BCD \\
&= B(\bar{C}\bar{D} + CD) + \bar{B}C\bar{D} + B\bar{C}D + ACD + \bar{A}B\bar{C}\bar{D} + \bar{A}\bar{B}CD \\
&= B + \bar{B}C\bar{D} + B\bar{C}D + ACD + \bar{A}B\bar{C}\bar{D} + \bar{A}\bar{B}CD \\
&= (B + \bar{B}C\bar{D}) + (B + B\bar{C}D) + (B + \bar{A}B\bar{C}\bar{D}) + ACD + \bar{A}\bar{B}CD \\
&= B + C\bar{D} + ACD + (\bar{A} + \bar{B})CD \\
&= (B + \bar{B}CD) + (ACD + \bar{A}CD) + C\bar{D} \\
&= B + CD + CD + C\bar{D} \\
&= B + C(D + \bar{D}) \\
&= B + C
\end{aligned}$$

2021 秋《计算机硬件基础》

03_1 组合逻辑 作业

一、填空题

- 1、从结构看，组合逻辑电路由门电路构成，不含反馈电路，也不含存储电路，信号从输入开始单向传输到输出。对于组合逻辑电路，任何时刻电路的输出仅由当时的输入信号决定。
- 2、将加在电路若干输入端中的某一个输入端的信号变换成相应的一组二进制代码输出的过程叫做编码。
- 3、将二进制代码所表示的信息翻译成对应输出的高低电平信号的过程称为译码； n 位二进制译码器有 n 个输入，有 2^n 个输出，工作时译码器只允许有一个输出有效。
- 4、输出低电平有效的二十进制译码器的输入 8421BCD 码 $A_3 \sim A_0$ 为 0110 时，其输出 $\bar{Y}_0 \sim \bar{Y}_9 =$ 1111110111。

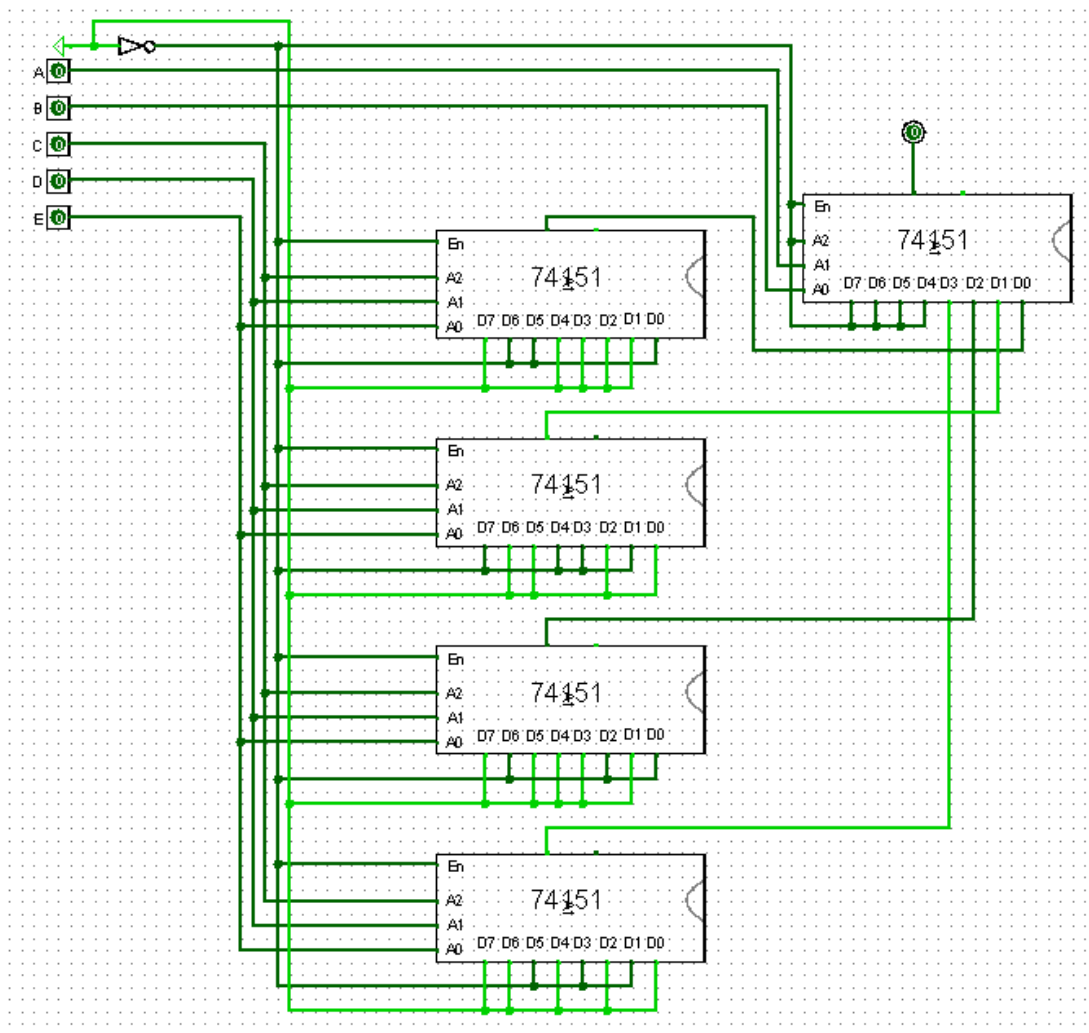
二、选择题

- 1、组合逻辑电路的竞争-冒险是由于（ C ）引起的。
A、电路不是最简
B、电路有多个输出
C、构成电路的逻辑元件存在传输延迟
D、电路使用不同的门电路
- 2、能实现从多个输入端中选出一路作为输出的电路称为（ C ）。
A、触发器
B、计数器
C、数据选择器
D、译码器
- 3、只考虑本位数而不考虑低位来的进位的加法称为（ B ）。
A、全加
B、半加
C、全减
D、半减
- 4、如需要判断两个二进制数的大小或相等，可以使用（ D ）电路。
A、译码器
B、编码器
C、数据选择器
D、数值比较器

三、问答与计算题

说明：分析与计算题要求写出分析推导过程，给出必要的公式。

- 1、用 8 选 1 数据选择器 CT74151 实现下列函数：
$$F(A,B,C,D,E) = \sum m(1,2,3,4,7,8,10,13,14,17,19,20,21,23,24,26,28,30,31)$$



实现思路：

首先搭建 32 位的选择器，将低 3 位用 C,D,E 进行选择也即划分为 0~7, 8~15, 16~23, 24~31 四部分，之后对于高 2 位用 A,B 进行选择（此时只需要 4 路选择器，因此将 74151 的高 4 位置 0）。之后，在 32 位的选择器的基础上，筛选出不为真值的输入置 0，为真值的输入置 1 即可。

2、在三进制数系统中，存在三个数字：0, 1, 2。表 1 定义了一个三进制数的半加器。

（1）设计一个实现此半加器的电路。要求用二进制编码表示三进制数，例如每个三进制数用 2 位表示。令 $A = a_1a_0$, $B = b_1b_0$, $Sum = s_1s_0$, 进位信号 Carry 是二进制信号。编码方案为：00 = (0)₃, 01 = (1)₃, 10 = (2)₃。要求电路的成本最低。

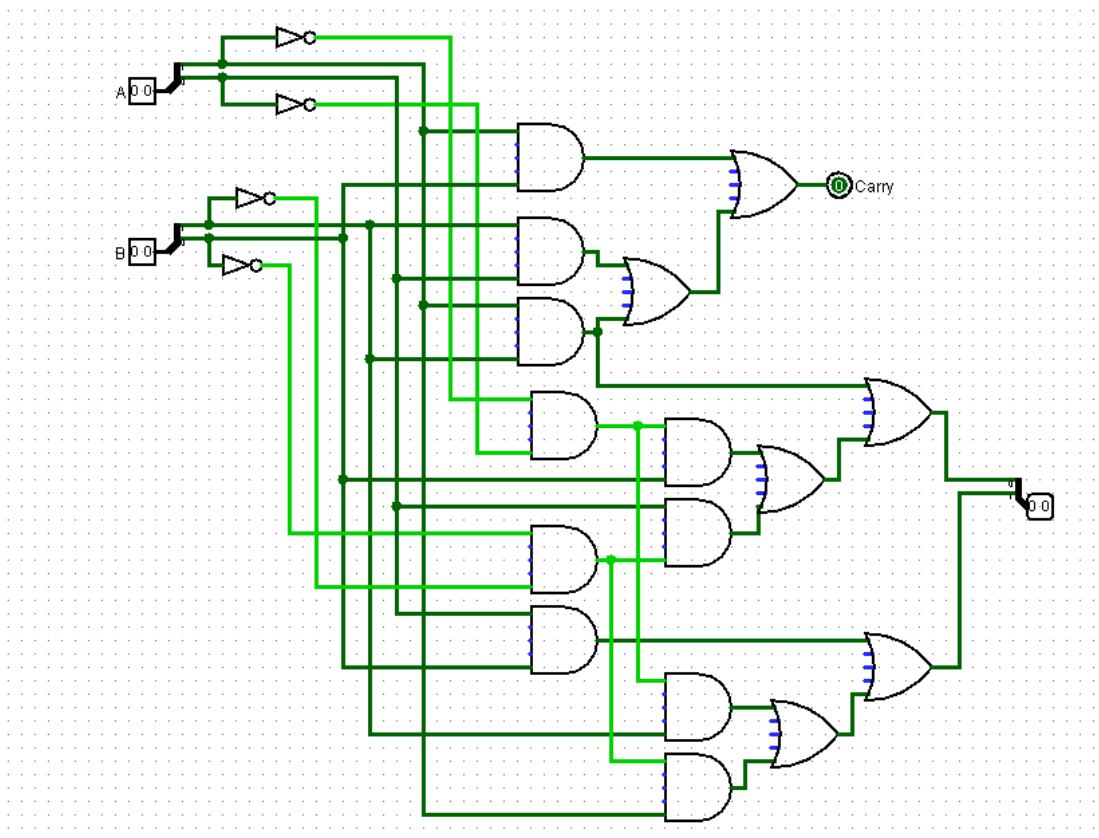
（2）使用上述描述的方法，设计一个三进制全加器电路。

表 1 三进制半加器

A	B	Carry	Sum
0	0	0	0
0	1	0	1
0	2	0	2
1	0	0	1
1	1	0	2

1	2	1	0
2	0	0	2
2	1	1	0
2	2	1	1

(1)



由真值表化简得到:

$$\text{Carry} = a_1b_1 + a_1b_0 + a_0b_1$$

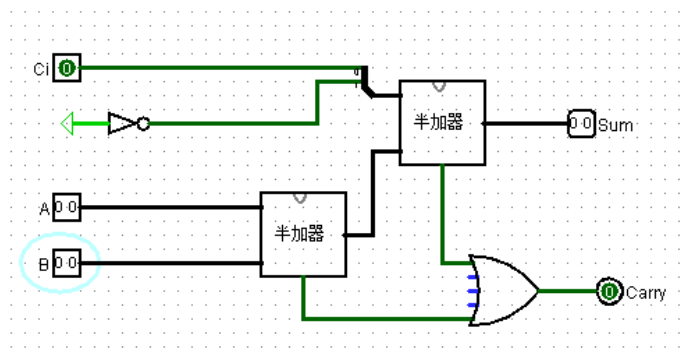
$$s_1 = !a_0!a_1b_1 + a_1!b_0!b_1 + a_0b_0$$

$$s_0 = !a_0!a_1b_0 + a_0!b_0!b_1 + a_1b_1$$

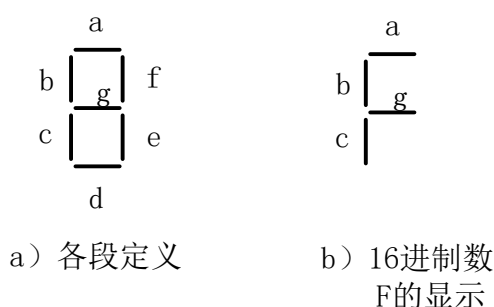
从而搭建上述电路

(2)

利用上述半加器，可以通过两个半加器得到一个全加器



3、7 段数码管是由 7 个独立的发光管构成的，每个发光管有一个驱动控制信号。当驱动控制信号为高电平(逻辑 1)时，则信号对应的发光管发光。现需设计 7 段数码管的控制电路，使之能够根据 4 位输入 $x[3:0]$ 显示 0~9, A~F 共 16 个图案。7 段数码管控制电路输出信号为各数码管的驱动控制信号，即 a, b, c, d, e, f, g。数码管各段的定义和 16 进制数“F”（对应 abcdefg 的二进制输出为 1110001）的显示如下图所示。



- (1) 请给出 7 段数码管控制电路的输入输出信号真值表。
- (2) 根据真值表写出各输出信号的逻辑表达式，并化简。

(1)

x_3	x_2	x_1	x_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	0	0	0	1	1	0
0	0	1	0	1	0	1	1	0	1	1
0	0	1	1	1	0	0	1	1	1	1
0	1	0	0	0	1	0	0	1	1	1
0	1	0	1	1	1	0	1	1	0	1
0	1	1	0	1	1	1	1	1	0	1
0	1	1	1	1	0	0	0	1	1	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1	1	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	1	1	1	1	0	1
1	1	0	0	1	1	1	1	0	0	0
1	1	0	1	0	0	1	1	1	1	1
1	1	1	0	1	1	1	1	0	0	1
1	1	1	1	1	1	1	0	0	0	1

(2)

$$\begin{aligned}
 a &= m_0 + m_2 + m_3 + m_5 + m_6 + m_7 + m_8 + m_9 + m_{10} + m_{12} + m_{14} + m_{15} = (!x_3!x_2!x_1!x_0) \\
 &+ (!x_3!x_2x_1!x_0) + (!x_3!x_2x_1x_0) + (!x_3x_2!x_1!x_0) + (!x_3x_2x_1!x_0) + (!x_3x_2x_1x_0) + (x_3!x_2!x_1!x_0) \\
 &+ (x_3!x_2!x_1x_0) + (x_3!x_2x_1!x_0) + (x_3x_2!x_1!x_0) + (x_3x_2x_1!x_0) + (x_3x_2x_1x_0) \\
 b &= m_0 + m_4 + m_5 + m_6 + m_8 + m_9 + m_{10} + m_{11} + m_{12} + m_{14} + m_{15} = (!x_3!x_2!x_1!x_0) + \\
 &(!x_3x_2!x_1!x_0) + (!x_3x_2!x_1x_0) + (!x_3x_2x_1!x_0) + (x_3!x_2!x_1!x_0) + (x_3!x_2!x_1x_0) + (x_3!x_2x_1!x_0) \\
 &+ (x_3!x_2x_1x_0) + (x_3x_2!x_1!x_0) + (x_3x_2x_1!x_0) + (x_3x_2x_1x_0) \\
 c &= m_0 + m_2 + m_6 + m_8 + m_{10} + m_{11} + m_{12} + m_{13} + m_{14} + m_{15} = (!x_3!x_2!x_1!x_0) + \\
 &(!x_3!x_2x_1!x_0) + (!x_3x_2x_1!x_0) + (x_3!x_2!x_1!x_0) + (x_3!x_2x_1!x_0) + (x_3!x_2x_1x_0) + (x_3x_2!x_1!x_0) +
 \end{aligned}$$

$$(x_3x_2!x_1x_0) + (x_3x_2x_1!x_0) + (x_3x_2x_1x_0)$$

$$d = m_0 + m_2 + m_3 + m_5 + m_6 + m_8 + m_9 + m_{11} + m_{12} + m_{13} + m_{14} = (!x_3!x_2!x_1!x_0) + (!x_3!x_2x_1!x_0) + (!x_3!x_2x_1x_0) + (!x_3x_2!x_1x_0) + (!x_3x_2x_1!x_0) + (x_3!x_2!x_1!x_0) + (x_3!x_2!x_1x_0) + (x_3!x_2x_1x_0) + (x_3x_2!x_1!x_0) + (x_3x_2!x_1x_0) + (x_3x_2x_1!x_0)$$

$$e = m_0 + m_1 + m_3 + m_4 + m_5 + m_6 + m_7 + m_8 + m_9 + m_{10} + m_{11} + m_{13} = (!x_3!x_2!x_1!x_0) + (!x_3!x_2!x_1x_0) + (!x_3!x_2x_1x_0) + (!x_3x_2!x_1!x_0) + (!x_3x_2!x_1x_0) + (!x_3x_2x_1!x_0) + (!x_3x_2x_1x_0) + (x_3!x_2!x_1!x_0) + (x_3!x_2!x_1x_0) + (x_3!x_2x_1!x_0) + (x_3!x_2x_1x_0) + (x_3x_2!x_1x_0)$$

$$f = m_0 + m_1 + m_2 + m_3 + m_4 + m_7 + m_8 + m_9 + m_{10} + m_{13} = (!x_3!x_2!x_1!x_0) + (!x_3!x_2!x_1x_0) + (!x_3!x_2x_1!x_0) + (!x_3!x_2x_1x_0) + (!x_3x_2!x_1!x_0) + (!x_3x_2x_1!x_0) + (x_3!x_2!x_1!x_0) + (x_3!x_2!x_1x_0) + (x_3!x_2x_1!x_0) + (x_3x_2!x_1x_0)$$

$$g = m_2 + m_3 + m_4 + m_5 + m_6 + m_8 + m_9 + m_{10} + m_{11} + m_{13} + m_{14} + m_{15} = (!x_3!x_2x_1!x_0) + (!x_3!x_2x_1x_0) + (!x_3x_2!x_1!x_0) + (!x_3x_2!x_1x_0) + (!x_3x_2x_1!x_0) + (x_3!x_2!x_1!x_0) + (x_3!x_2!x_1x_0) + (x_3!x_2x_1!x_0) + (x_3!x_2x_1x_0) + (x_3x_2!x_1x_0) + (x_3x_2x_1!x_0) + (x_3x_2x_1x_0)$$

化简:

$$a = !x_2!x_0 + !x_3x_1 + !x_3x_2x_0 + x_2x_1 + x_3!x_2!x_1 + x_3!x_0$$

$$b = !x_1!x_0 + !x_3x_2!x_1 + x_2!x_0 + x_3!x_2 + x_3x_1$$

$$c = !x_2!x_0 + x_1!x_0 + x_3x_1 + x_3x_2$$

$$d = !x_3!x_2!x_0 + !x_2x_1x_0 + x_2!x_1x_0 + x_2x_1!x_0 + x_3!x_1$$

$$e = !x_3!x_1 + !x_3x_0 + !x_1x_0 + !x_3x_2 + x_3!x_2$$

$$f = !x_3!x_2 + !x_3!x_1!x_0 + !x_2!x_0 + !x_3x_1x_0 + x_3!x_1x_0$$

$$g = !x_2x_1 + x_1!x_0 + !x_3x_2!x_1 + x_3!x_2 + x_3x_0$$

2021 秋《计算机硬件基础》

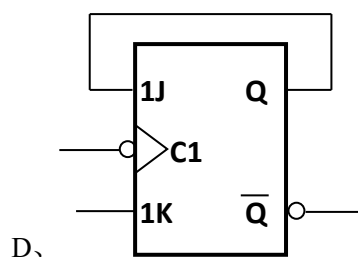
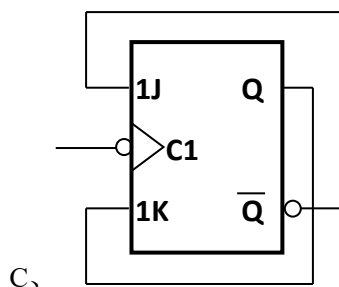
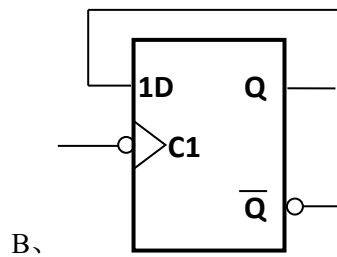
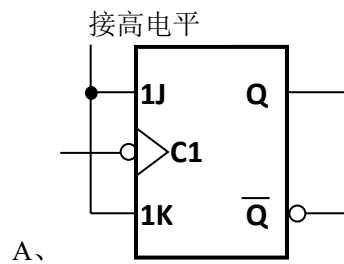
03_2 时序逻辑 作业

一、 填空题

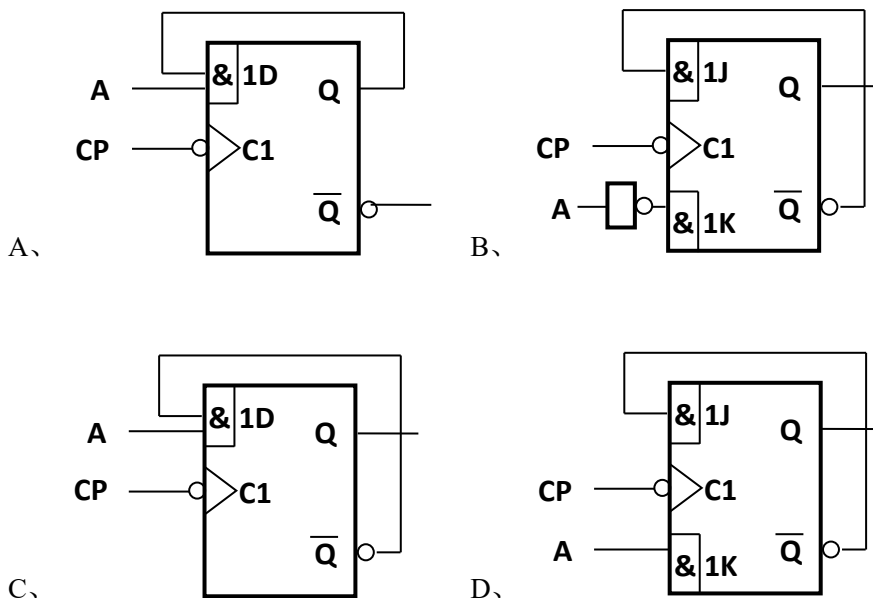
- 1、时序逻辑电路由组合逻辑电路和存储电路两部分组成。
- 2、由与非门构成的基本 RS 触发器约束条件是 $RS=0$ 。
- 3、钟控 JK 触发器的特性方程为 $Q^{n+1} = J\bar{Q}^n + \bar{K}Q^n$ 。
- 4、时序逻辑电路按触发器时钟端的连接方式不同可以分为同步时序逻辑和异步时序逻辑两类。
- 5、四级移位寄存器可以存放4位二进制数据。
- 6、集成计数器的模值是固定的，但可以用反馈复位法法和预置法来改变它们的模值。
- 7、由 8 级触发器构成的十进制计数器模值为100。
- 8、通过级联方法，把三片 4 位二进制计数器 CT74161 连接成为 12 位二进制计数器后，其最大模值是4096。
- 9、根据在 CP 控制下，逻辑功能的不同，常把时钟触发器分为RS 触发器、D 触发器和JK 触发器等 3 种类型。
- 10、时序逻辑电路的功能表示方法有状态转换表、状态转换图、状态方程和时序图。

二、 选择题

- 1、若 JK 触发器的原始状态为 0，欲在 CP 作用后保持 0 状态，则激励函数 JK 的值应是（ C ）。
A、J=1, K=1 B、J=0, K=0
C、J=0, K=x D、J=x, K=x
- 2、下列电路中，只有（ D ）不能实现 $Q^{n+1} = \bar{Q}^n$ 。



3、如下各触发器电路中，能实现 $Q^{n+1} = \overline{Q}^n + AQ^n$ 功能的电路是（ B ）。



4、用 3 级触发器可以记忆（ A ）种不同的状态。

- A、8 B、16
C、128 D、256

5、同步计数器是指（ B ）的计数器。

- A、由同类型的触发器构成
B、各触发器时钟端连在一起，统一由系统时钟控制
C、可用前级的输出做后级触发器的时钟
D、可用后级的输出做前级触发器的时钟

6、由 4 级触发器构成的二进制计数器，其模值为（ A ）。

- A、16 B、20
C、1000 D、1024

7、同步 4 位二进制减法计数器的借位方程是 $B = \overline{Q_4} \overline{Q_3} \overline{Q_2} \overline{Q_1}$ ，则可知 B 的周期和正脉冲宽度为（ B ）。

- A、16 个 CP 周期和 2 个 CP 周期
B、16 个 CP 周期和 1 个 CP 周期
C、8 个 CP 周期和 8 个 CP 周期
D、8 个 CP 周期和 4 个 CP 周期

8、已知 $Q_3 Q_2 Q_1 Q_0$ 是同步十进制计数器的触发器输出，若以 Q_3 作进位 C，则 C 的周期和正脉冲宽度是（ B ）。

- A、10 个 CP 脉冲，正脉冲宽度为 1 个 CP 周期
B、10 个 CP 脉冲，正脉冲宽度为 2 个 CP 周期
C、8 个 CP 脉冲，正脉冲宽度为 1 个 CP 周期
D、8 个 CP 脉冲，正脉冲宽度为 2 个 CP 周期

9、一个 4 位移位寄存器原来的状态为 0000，如果串行输入始终为 1，则经过 4 个移位脉冲后寄存器的内容为（ D ）。

- A、0001 B、0111
C、1110 D、1111
- 10、 可以用来实现并/串转换和串/并转换的器件是（ B ）。
- A、计数器 B、移位寄存器
C、存储器 D、全加器
- 11、 设计模值为 36 的计数器至少需要（ B ）级触发器。
- A、5 B、6
C、3 D、4
- 12、 用 Verilog HDL 设计同步清除的计数器时，在 `always` 语句的敏感参数表中（ A ）。
- A、需要列出时钟信号和清除信号标示符的有效边沿
B、只需要列出时钟信号标示符的有效边沿
C、只需要列出清除信号标示符的有效边沿
D、只需要列出时钟信号或者清除信号标示符的有效边沿

三、 分析与设计

- 1、有一简单时序逻辑电路如图 3.1 所示，试写出当 $C=0$ 和 $C=1$ 时，电路的状态方程 Q^{n+1} ，并说出各自实现的功能。

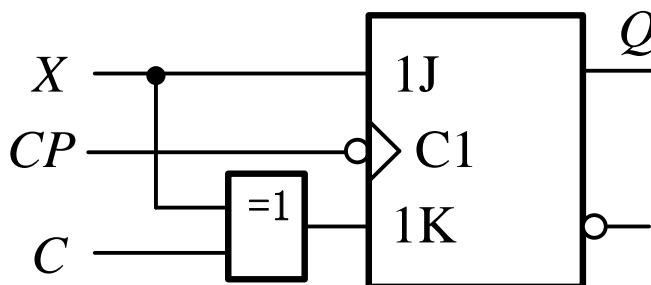


图 3.1

$C=0$ 时：

$$Q^{n+1} = J\bar{Q}^n + \bar{K}Q^n = X\bar{Q}^n + \bar{X}Q^n = X \oplus Q^n$$

实现的功能：X 与 Q^n 取异或

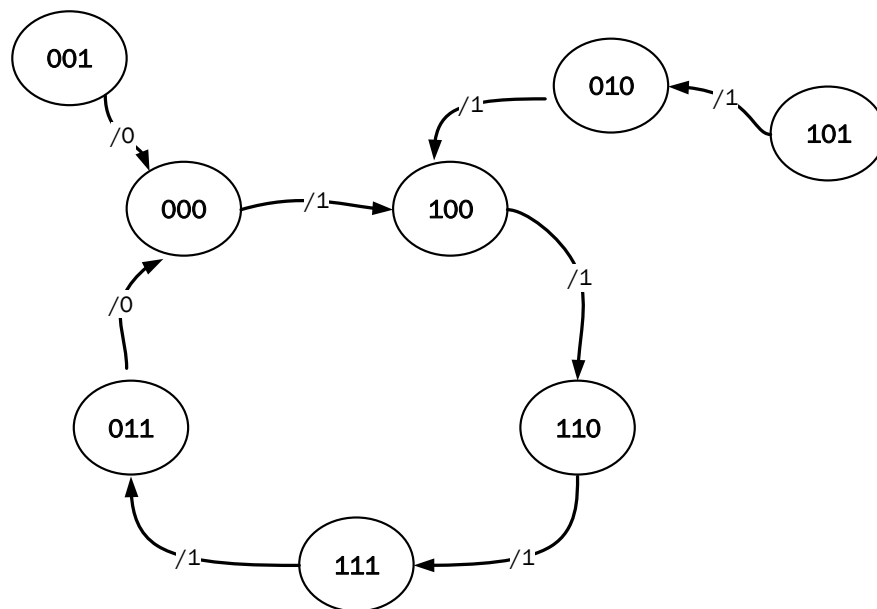
$C=1$ 时：

$$Q^{n+1} = J\bar{Q}^n + \bar{K}Q^n = X\bar{Q}^n + XQ^n = X$$

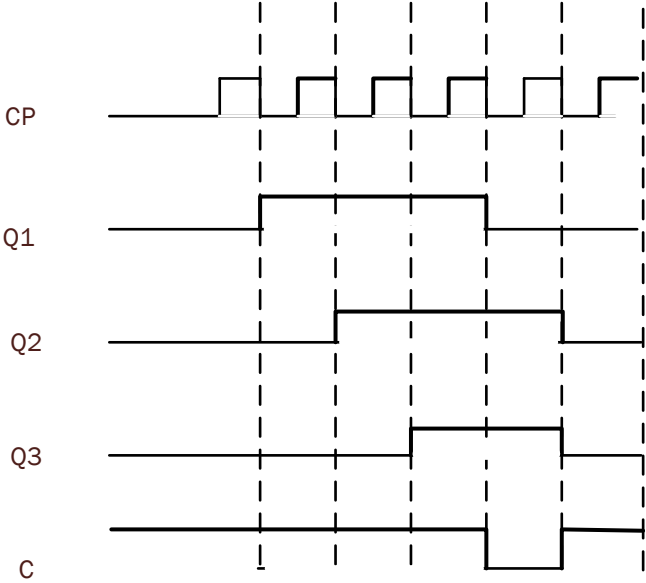
实现的功能：直接置为 X

- 2、分析下图所示电路，要求：

- 1) 写出分析过程，包括各级触发器的驱动方程和状态方程；
- 2) 画出状态转换表、状态转换图和时序图；
- 3) 说明电路特点。



时序图



3)
电路特点：具有自启动特性的同步五进制加法计数器

3、集成 4 位二进制计数器 CT74161 的逻辑符号如图 3.3 所示，其功能表如表 3.1 所示，触发器输出低位到高位次序是 Q_0 至 Q_3 ，输出 $C = ETQ_3^n Q_2^n Q_1^n Q_0^n$ 。试用一片 CT74161 采用输出 C 预置法实现十二进制计数器，画出电路连接图。

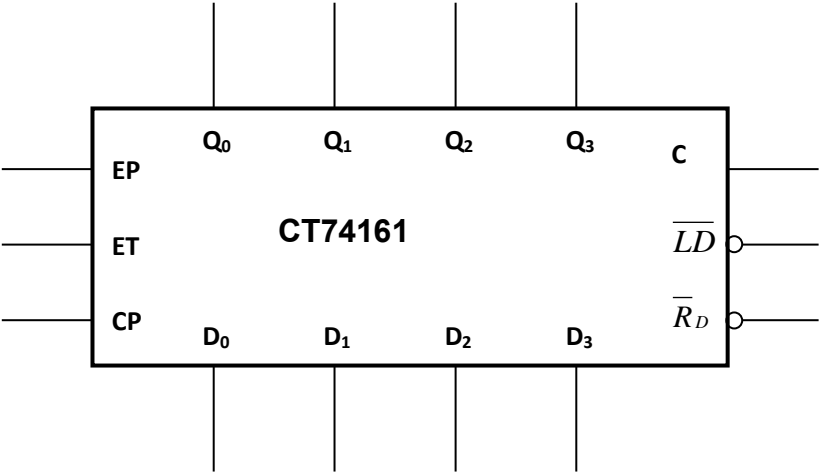


图 3.3 4 位二进制计数器 CT74161 的逻辑符号

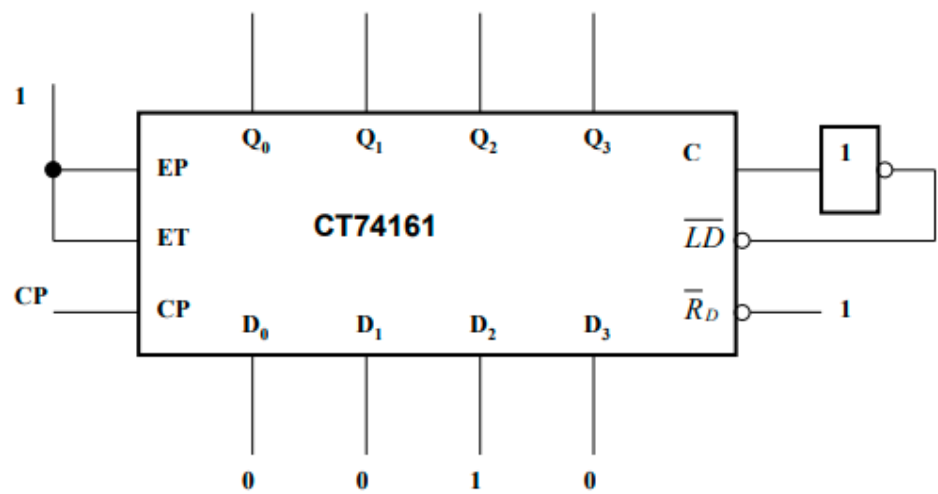
表 3.1 CT74161 的功能表

\overline{R}_D	\overline{LD}	EP	ET	CP	功能
------------------	-----------------	----	----	----	----

0	×	×	×	×	复位
1	0	×	×	↑	预置
1	1	0	0	↑	保持
1	1	0	1	↑	保持
1	1	1	0	↑	保持
1	1	1	1	↑	计数

输出 C 预置法:

$$(16 - 12)_{10} = (4)_{10} = (0100)_2 = D_3D_2D_1D_0$$

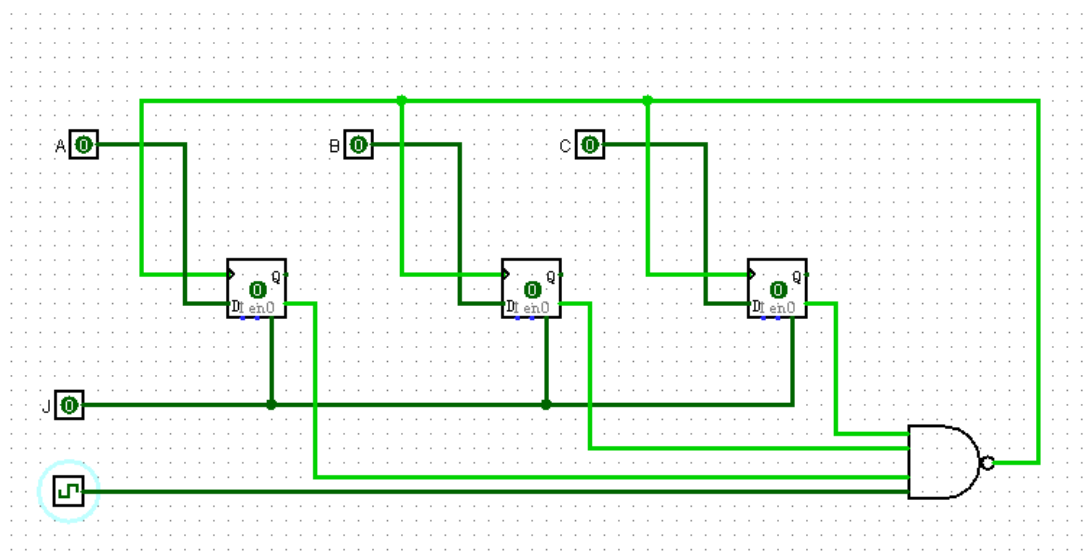


4、使用 D 触发器和与非门设计一个 3 人抢答逻辑电路，给出逻辑电路图并叙述电路工作过程。对电路的具体要求如下：

- 1) 每个参赛者（A、B 和 C）控制一个按钮，用按钮发出抢答信号。
- 2) 竞赛主持人 J 控制另一个按钮，用于电路复位。
- 3) 竞赛开始后，先按动按钮者将对应的一个发光二极管点亮，此后其他 2 人再按动按钮对电路不起作用。

提示：抢答逻辑电路通常用于智力竞赛的抢答比赛中。由于参赛者按动按钮发出的信号不能自行保持，而且按动的动作可能有先后、长短之别，所以需要 3 个触发器分别保存 3 个参赛者按动按钮发出的信号。由于只要求触发器具有置 1（抢答）、置 0（复位）功能即可，所以采用 RS、D、JK 触发器均可，对结构类型也无特定要求。

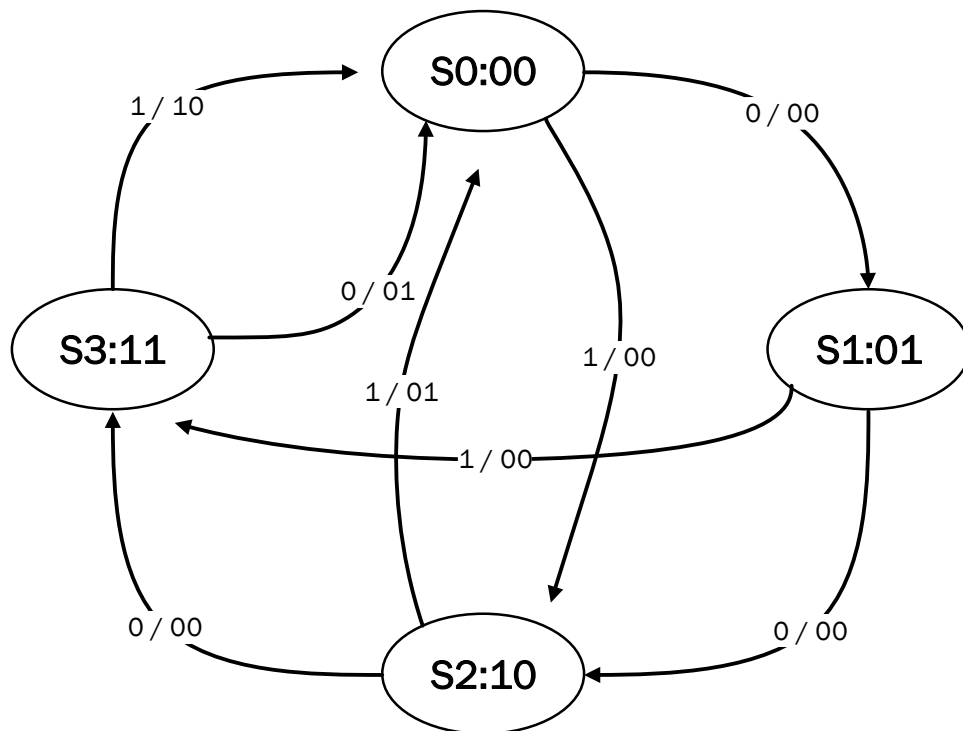
此外，最先产生的抢答信号还应控制其它后来产生的抢答信号无法改变自身触发器的值，这样在抢答结束时就可根据各触发器的状态判断是哪位选手最先按下抢答按钮的。因此，需要将 3 个触发器的反相输出 \overline{Q} 分别引入到一个与非门的输入端，并与输入时钟信号相与，再将与非门的输出作为各触发器的时钟信号。



电路工作过程：当 A、B、C 任何一个按钮首先按下时，对应的触发器被置 1；这个触发器的 \overline{Q} 端随之变为低电平，将与非门 G 封锁；其余的触发器不再有 CP 信号输入，无法再置 1。

5、设计一个自动售货机控制器，每次可以任意投入一枚 1 分或 2 分的硬币。货物价格为 4 分，当投入足够的钱后，售货机吐出货物并找零钱。请完成下列任务：

- 1) 画出实现上述功能的状态机；
 - 2) 列出二进制编码的状态转换表和输出逻辑真值表，给出次态每一位编码的逻辑函数表达式和输出逻辑函数表达式，并化简。
- 1) 可实现 Mealy 型状态机如下



状态：

S0:00 未投入硬币

S1:01 已投入 1 分

S2:10 已投入 2 分

S3:11 已投入 3 分

输入：

I0:0 投入 1 分

I1:1 投入 2 分

输出：

O0:00 无操作

O1:01 出货

O2:10 出货并找零 1 分

2)

状态转换：

s1	s0	i	ns1	ns0
0	0	0	0	1
0	0	1	1	0
0	1	0	1	0
0	1	1	1	1
1	0	0	1	1
1	0	1	0	0
1	1	0	0	0
1	1	1	0	0

$$ns1 = !s1!s0i + !s1s0!i + !s1s0i + s1!s0!i = !s1s0 + !s1!s0i + s1!s0!i$$
$$ns0 = !s1!s0!i + !s1s0i + s1!s0!i = !s0!i + !s1s0i$$

输出逻辑：

s1	s0	i	o1	o0
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	0

$$o1 = s1s0i$$
$$o0 = s1!s0i + s1s0!i$$

2021 秋《计算机硬件基础》

04 指令系统和汇编语言 作业

一、 填空

- 1、将 10 进制数 35 转换为 8 位二进制数是 00100011。
- 2、将二进制数 00010101 转换为 16 进制数是 0x15。
- 3、将 10 进制数-35 转换为 8 位二进制补码是 11011101。
- 4、将 8 进制数 204 转换为 10 进制数是 132。
- 5、请判断以下两个补码表示的二进制数做二进制加法后是否溢出： 溢出。

01101110
00011010
- 6、对 8 位 16 进制数 0x88 做符号扩展成 16 位数是：0x ff88。
- 7、下列代码段存储在内存中，起始地址为 0x00012344，分支指令执行后 PC 的两个可能的值分别是：0x 00012344 和 0x 00012354。同时，请在注释位置用伪代码形式对每条指令做出描述。

```
loop:lw$t0, 0($a0)    # t0 = memory[a0 + 0]  
      addi$a0, $a0, 4  # a0 = a0 + 4  
      andi$t1, $t0, 1  # t1 = t0 and 1  
      beqz$t1, loop    # if t1 == 0: go to loop
```

二、 简答与设计

- 1、将下列汇编语言指令翻译成机器语言代码，以 16 进制表示。

```
loop:addu$a0, $0, $t0    # 0x00082021  
      ori$v0, $0, 4      # 0x34020004  
      syscall            # 0x0000000c  
      addi$t0, $t0, -1   # 0x2108ffff  
      bnez$t0, loop      # 0x1500fffb  
      andi$s0, $s7, 0xffc0 # 0x32f0ffc0  
      or$a0, $t7, $s0    # 0x01f02025  
      sb$a0, 4($s6)      # 0xa2c40004
```

```
srl$s7, $s7, 4      # 0x0017b902
```

2、写一个 MIPS 汇编程序，要求对内存以“example100”为标签（label）的数据段中前 100 个字（words）的数据求和，并将结果存入紧跟在这 100 个字之后的内存中。

```
addi $t0 $zero 0
addi $t1 $zero 0
Loop:
    lw $t2 example100($t0)
    add $t1 $t1 $t2
    addi $t0 $t0 4
    slti $t3 $t0 400
    beq $t3 1 Loop
sw $t1 example100($t0)
```

3、写一段 MIPS 汇编语言代码，将内存中“SRC”标签开始的 100 个字的一块数据转移到内存中另一块以“DEST”标签开始的空间中。

```
addi $t0 $zero 0
Loop:
    lw $t2 SRC($t0)
    sw $t2 DEST($t0)
    addi $t0 $t0 4
    slti $t3 $t0 400
    beq $t3 1 Loop
```

4、写一个 MIPS 函数 ABS，通过\$a0 传入一个 32 位整数，将这个数的绝对值存回\$a0。再写一段主程序，调用两次 ABS 并输出结果，每次传给 ABS 的数不同。


```

.text
main:
    addi $a0 $zero -10
    jal ABS
    li $v0 1 #输出 $a0 的值
    syscall
    addi $a0 $zero 10
    jal ABS
    li $v0 1 #输出 $a0 的值
    syscall
    li $v0 10 #结束程序
    syscall

ABS:
    slti $t0 $a0 0
    beq $t0 0 returnABS
    sub $a0 $zero $a0
returnABS:
    jr $ra

```

5、考虑如下 32-bit x86 机器代码：

```

int fact(int n)
{
    if (n == 1)
        return n;
    else
        return n * fact(n - 1);
}

```

```

080483a4 <fact>:
80483a4: 55                push    %ebp
80483a5: 89 e5            mov     %esp, %ebp
80483a7: 53              push    %ebx
80483a8: 83 ec 04        sub     $0x4, %esp
80483ab: 8b 5d 08        mov     0x8(%ebp), %ebx
80483ae: 83 fb 01        cmp     $0x1, %ebx
80483b1: 74 0e          je      80483c1 <fact+0x1d>
80483b3: 8d 43 ff        lea     0xffffffff(%ebx), %eax
80483b6: 89 04 24        mov     %eax, (%esp)
80483b9: e8 e6 ff ff ff  call    80483a4 <fact>
80483be: 0f af d8        imul    %eax, %ebx
80483c1: 89 d8          mov     %ebx, %eax
80483c3: 83 c4 04        add     $0x4, %esp
80483c6: 5b             pop     %ebx
80483c7: 5d             pop     %ebp
80483c8: c3             ret

```

A. 填补栈的空白，从 caller 调用 fact(4) 开始，在调用 fact(2) 的 call 指令前立即结束。

已知：

- 调用 fact(4) 的函数的返回地址：0x080483e6
- fact(4) 被调用时 %ebp 值为 0xffffd848
- Callee（被调用者）保存的寄存器有：%ebx, %esi, %edi

B. 调用 fact(2) 的 call 指令执行前，%ebp 最后的值是多少？

C. 调用 fact(2) 的 call 指令执行前，%esp 最后的值是多少？

Stack address	The diagram starts with the argument for fact(4)	
	+-----+	
0xffffd830		
	+-----+	
0xffffd82c		
	+-----+	
0xffffd828		
	+-----+	
0xffffd824		
	+-----+	
0xffffd820		
	+-----+	
0xffffd81c		
	+-----+	
0xffffd818		
	+-----+	
0xffffd814		
	+-----+	
0xffffd810		
	+-----+	

B.

C.

6、下面代码中，M 和 N 为 #define 声明的常数，它们的值分别是多少？为什么？

```
int array1[M][N];
int array2[N][M];
void copy(int i, int j) {
    array1[i][j] = array2[j][i];
}
```

由以上代码生成的汇编代码为：

```
copy:
    pushl %ebp
    movl %esp,%ebp
    pushl %ebx
    movl 8(%ebp),%ecx
    movl 12(%ebp),%eax
    leal 0(,%eax,4),%ebx
    leal 0(,%ecx,8),%edx
    subl %ecx,%edx
    addl %ebx,%eax
    sall $2,%eax
    movl array2(%eax,%ecx,4),%eax ; int array2[N][M];
    movl %eax,array1(%ebx,%edx,4) ; int array1[M][N];
    popl %ebx
    movl %ebp,%esp
    popl %ebp
    ret
```

1、



根据如下指令回答下列 3 个问题。

- (1) 对上述指令而言，图 1 中的控制单元要产生哪些控制信号？
- (2) 对上述指令而言，要用到哪些功能单元？
- (3) 哪些功能单元会产生输出，但输出不会被以上指令用到？对以上指令而言，哪些功能单元不产生任何输出？

不同单元有不同的延迟时间。在图 1 中有七种主要单元。对一条指令而言，关键路径(产生最长延迟的那条路径)上各个单元的延迟时间决定了该指令的最小延迟。假设个单元的延迟时间如下表所示，回答下列 3 个问题。

	指令存储器	加法器	多选器	ALU	寄存器堆	数据存储器	控制
a.	400ps	100ps	30ps	120ps	200ps	350ps	100ps
b.	500ps	150ps	100ps	180ps	220ps	1000ps	65ps

(4)对一条 MIPS 的与指令(AND)而言, 关键路径是什么?

(5)对一条 MIPS 的装载指令(LW)而言, 关键路径是什么?

(6)对一条 MIPS 的相等则分支指令(BEQ)而言, 关键路径是什么?

(1) 对上述指令而言, 图 1 中的控制单元要产生哪些控制信号

	RegWrite	ALUSrc	MemRead	MemWrite	ALUop	MemReg	Branch
a.	1	0(Reg)	0	0	0(Add)	0(ALU)	0
b.	1	1(Imm)	1	0	0(Add)	1(Mem)	0

(2) 对上述指令而言, 要用到哪些功能单元

a.除 Data memory 和分支加法器以外所有单元

b.除分支加法器意外所有单元

(3) 哪些功能单元会产生输出, 但输出不会被以上指令用到

a. 分支加法器

b. 分支加法器 寄存器堆 Rt 输出端

哪些功能单元不产生任何输出

a. Data memory

b. 无

(4) 对一条 MIPS 的与指令(AND)而言, 关键路径是什么

指令存储器 寄存器堆 多选器 ALU 多选器 寄存器堆

(5) 对一条 MIPS 的装载指令(LW)而言, 关键路径是什么

指令存储器 寄存器堆 多选器 ALU 数据存储器 多选器 寄存器堆

(6) 对一条 MIPS 的相等则分支指令(BEQ)而言, 关键路径是什么

指令存储器 寄存器堆 多选器 ALU 多选器

2、图 1 中基本的单周期 MIPS 实现仅能实现某些指令。可以在这个指令集中加入新的指令,

但决定是否加入取决于给处理器的数据通路和数据通路增加的复杂度。对于下表中的新指令

而言, 试回答下列 3 个问题。

	指令	解释
a.	add3 Rd,Rs,Rt,Rx	Reg[Rd]=Reg[Rs]+Reg[Rt]+Reg[Rx]
b.	sll Rt,Rd,Shift	Reg[Rd]=Reg[Rt]<<Shift(左移)

(1)对上述指令而言, 哪些已有的单元还可以被使用?

(2)对上述指令而言, 还需要增加哪些功能单元?

(3)为了支持这些指令, 需要在控制单元增加哪些信号?

当设计者考虑改进处理器数据通路时，往往要考虑性能与成本的折中。假设我们从图 1 的数据通路出发，其中指令存储器(Instruction Memory)、加法器(Add)、多选器(Mux)、ALU、寄存器堆(Registers)、数据寄存器(Data Memory)和控制单元(Control)的延迟分别为 400ps、100ps、30ps、120ps、200ps、350ps 和 100ps，相应的成本分别为 1000、30、10、100、200、2000 和 500。试根据表中的改进分别回答下列问题。

	改进	延迟	成本	优势
a.	更快的加法器	加法单元-20ps	每个加法单元+20	把已有的加法器用更快的加法器替代
b.	更大的寄存器堆	寄存器堆+100ps	寄存器堆+200	需要更少的load和store指令。这将导致指令数减少5%

(4)改进前后的时钟周期分别是多少？

(5)改进后将获得多大的加速比？

(6)比较改进前后的性能/价格比，进行这样的改进是否有意义？

(1) 对上述指令而言，哪些已有的单元还可以被使用

除分支加法器、Data memory 之外的所有单元

(2) 对上述指令而言，还需要增加哪些功能单元

- 寄存器堆 增加 Rx 读入和输出端口；在 ALU 中增加一个输入端
- 在 ALU 中增加左移功能

(3) 为了支持这些指令，需要在控制单元增加哪些信号

- ALUOp 增加选择 add3 的信号
- ALUOp 增加选择 sll 的信号

(4) 改进前后的时钟周期分别是多少

改进前：400+200+30+120+350+30+200=1330ps

改进后：a. 无变化

b. 400+300+30+120+350+30+300=1530ps

(5) 改进后将获得多大的加速比

- 加速比：1
- 加速比： $(1 \times 1330) / (0.95 \times 1530) = 0.915$

(6) 比较改进前后的性能/价格比，进行这样的改进是否有意义

改进前：1000+2×50+3×10+100+200+2000+500=3890

改进后：a. 1000+2×30+3×10+100+200+2000+500=3930 相对成本 3930 / 3890 = 1.010

b. 1000+2×30+3×10+100+400+2000+500=4090 相对成本 4090 / 3890 = 1.051

a. 性能/价格比 1 / 1.010 = 0.99

b. 性能/价格比 0.915 / 1.051 = 0.87

都小于 1，可见改进都无意义

```

graph LR
    PC[PC] -- Read address --> IM[Instruction memory]
    IM -- Instruction --> Add[Add]
    Add -- 4 --> Add
    PC -- 4 --> Add
    Add --> PC
  
```

The diagram illustrates the internal components and data flow of a processor:

- PC (Program Counter):** Provides the initial address for the instruction memory.
- Instruction Memory:** Receives the PC value to output the instruction. It also provides the **Read address** to the registers.
- Registers:** A set of registers (Read register 1, Read register 2, Write register) that store data. They receive **Read data 1** and **Read data 2** from the ALU and **Write data** from the data memory. The **Write register** is controlled by **RegWrite**.
- ALU (Arithmetic Logic Unit):** Performs operations on data from the registers. It receives **ALUSrc** (source register data) and **ALU operation** (from the instruction). The ALU result is used to update the **PC** (via an **Add** block) and to write back to the registers (via a **Mux** block). The ALU also outputs a **Zero** flag to the data memory.
- Shift left 2:** A shift register that takes the ALU result and shifts it left by 2 bits to calculate the branch target address.
- Sign-extend:** A block that takes a 16-bit value from the registers and extends it to 32 bits for use in the ALU.
- Data Memory:** Receives **Write data** from the registers and outputs **Read data** to the registers. It is controlled by **MemWrite** and **MemRead** signals. The **Address** for data memory is provided by the ALU result.
- Mux (Multiplexer):** Selects between different data sources (e.g., register data, ALU result, branch target) to be written back to the registers or used for other purposes.

下表给出了实现处理器数据通路的逻辑单元延迟。试根据下表的两种情况分别回答下列问题。

- (1) 如果处理器只需做连续取指这一件事(见图 2)，那么时钟周期是多少？
- (2) 考虑一个与图 3 类似的数据通路，但是假设处理器只需处理无条件相对跳转指令，时钟周期是多少？
- (3) 同样考虑一个与图 3 类似的数据通路，但这次假设只需处理有条件相对跳转指令，

那么时钟周期是多少？(请注意图 3 中 ALU 的零输出端不是与数据存储器连接，该输出与选择 PC 值来源的多选器的控制有关)（提示：图 3 中靠右侧的加法器延迟应当按照 ALU 来计算）

根据下表的两种数据通路的逻辑单元，分别回答下列问题。

	单元
a.	执行加4的加法器(对PC)
b.	数据存储器

- (4)哪些类型的指令需要该单元？
- (5)对哪些类型的指令而言，该单元位于关键路径上？
- (6)假设仅需支持 beq 指令和 add 指令，讨论该单元的延迟变化对处理器时钟周期的影响。假设其他单元的延迟不变。

(1) 如果处理器只需做连续取指这一件事(见图 2)，那么时钟周期是多少

- a. 400ps
- b. 500ps

(2) 假设处理器只需处理无条件相对跳转指令，那么时钟周期是多少

- a. $400+20+2+120+30=572ps$
- b. $500+90+20+180+100=890ps$

(3) 假设只需处理有条件相对跳转指令，那么时钟周期是多少

- a. $400+200+30+120+30=780ps$
- b. $500+220+100+180+100=1100ps$

(4) 哪些类型的指令需要该单元

- a. 全部指令
- b. 访存相关的指令 lw sw 等

(5) 对哪些类型的指令而言，该单元位于关键路径上

- a. 无
- b. 访存相关的指令

(6) 假设仅需支持 beq 指令和 add 指令，讨论该单元的延迟变化对处理器时钟周期的影响

- a. 对 beq 指令 关键路径 $400+200+30+120+30=780ps$
对 add 指令 关键路径 $400+200+30+120+30+200=980ps$

涉及 PC+4 路径 $100+120+30=250ps$

当 PC+4 部分延迟小于 830ps 时，不会造成影响；大于 830ps 时，会延长时钟周期

- b. beq 和 add 均不涉及访存，因此无影响

4、本题讨论数据通路中不同的单元延迟对整个数据通路时钟周期的影响，以及指令如何利用不同的数据通路单元。根据下面的两种延迟情况，分别回答下列问题。

	指令存储器	加法器	多选器	ALU	寄存器堆	数据存储器	符号扩展	左移两位
a.	400ps	100ps	30ps	120ps	200ps	350ps	20ps	0ps
b.	500ps	150ps	100ps	180ps	220ps	1000ps	90ps	20ps

- (1)如果仅需支持 ALU 类指令(如 add、and 等)，处理器的时钟周期是多少？
- (2)如果仅需支持 lw 类指令，时钟周期是多少？
- (3)如果必须支持 add、beq、lw 和 sw 指令，时钟周期是多少？

假设各类型指令所占比例如下表所示，试根据下表的两种情况分别回答下列问题。

	add	addi	not	beq	lw	sw
a.	30%	15%	5%	20%	20%	10%
b.	25%	5%	5%	15%	35%	15%

- (4)数据存储器平均用了多少时钟周期？
- (5)符号扩展电路的输入平均用了多少时钟周期？在未用到该输入的其他时间，符号扩展电路在做什么？
- (6)如果可以将数据通路上某个单元的延迟减少 10%，应该减少哪个单元的延迟？改进后整个处理器的加速比是多少？

(1) 如果仅需支持 ALU 类指令(如 add、and 等)，处理器的时钟周期是多少

a. $400+200+30+120+30+200 = 980\text{ps}$

b. $500+220+100+180+100+220 = 1320\text{ps}$

(2) 如果仅需支持 lw 类指令，时钟周期是多少

a. $400+200+30+120+350+30+200 = 1330\text{ps}$

b. $500+220+100+180+1000+100+220 = 2320\text{ps}$

(3) 如果必须支持 add、beq、lw 和 sw 指令，时钟周期是多少

a. 1330ps

b. 2320ps

(4) 数据存储器平均用了多少时钟周期

a. $20\%(\text{lw}) + 10\%(\text{sw}) = 30\%$

b. $35\%(\text{lw}) + 15\%(\text{sw}) = 50\%$

(5) 符号扩展电路的输入平均用了多少时钟周期

a. $15\%(\text{addi}) + 20\%(\text{beq}) + 20\%(\text{lw}) + 10\%(\text{sw}) = 65\%$

b. $5\%(\text{addi}) + 15\%(\text{beq}) + 35\%(\text{lw}) + 15\%(\text{sw}) = 70\%$

在未用到该输入的其他时间，符号扩展电路在做什么

仍然取低 16 位作扩展，但该数据不会应用

(6) 如果可以将数据通路上某个单元的延迟减少 10%，应该减少哪个单元的延迟

a. $400+200+30+120+350+30+200 = 1330\text{ps}$ 可以选择 指令存储器 或 寄存器堆

减少量 $400 \times 10\% = 40\text{ps}$ 新时钟周期为 1290ps

b. $500+220+100+180+1000+100+220 = 2320\text{ps}$ 应该减少 数据存储器 的延迟

减少量 $1000 \times 10\% = 100\text{ps}$ 新时钟周期为 2220ps

改进后整个处理器的加速比是多少

a. $1330 / 1290 = 1.031$

b. $2320 / 2220 = 1.045$

5、在制造硅芯片时，材料的缺陷和制造错误会导致电路失效。一个非常普遍的问题是一根线上的信号会对相邻线上的信号产生影响，这被称为串扰。有一类串扰问题是这样的，某些线上的信号为常值(如电源线)，该线附近的线也被固定为 0(stuck-at-0)或 1(stuck-at-1)。试根据下表的两种缺陷(信号来自图 4)分别回答下列问题。

	有问题的信号
a.	指令存储器，输出信号第7位
b.	控制单元，输出信号MemtoReg

(1)设这样测试处理器的缺陷：先给 PC、寄存器堆、数据和指令存储器中设置一些值(可以自己选择)，执行一条指令，然后读出 PC、寄存器堆和存储器中的值；最后检查这些值以判断处理器中是否存在缺陷。你能设计这样一个方案检查该信号上是否有固定为 0 缺陷吗？

(2)条件同第①问，但是这次检查固定为 1 缺陷。你能只设计一个测试方案同时检查固定为 0 缺陷和固定为 1 缺陷吗？如果可以，请解释如何实现；如果不能，请说明理由。

(3)如果我们知道一个处理器在该信号上有一个固定为 1 缺陷，它还能用吗？为了使这个处理器仍然可用，我们必须将原来能在正常 MIPS 处理器上运行的程序做一些变换，使之可以在这个处理器上运行。假设指令存储器和数据存储器都很大，足够容纳变换后的程序。提示：将因为该缺陷不能用的指令替换为一系列能用的指令，这一系列指令与原指令功能相同。

根据下表的缺陷分别回答下列问题。

	缺陷
a.	固定为1
b.	如果指令的第31~26位全为0，则固定为0，否则无缺陷

(4)条件同第①问，这次检测控制信号 MemRead 是否存在上表中的缺陷？

(5)条件同第①问，这次检测控制信号 Jump 是否存在上表中的缺陷？

(6)使用第①问中描述的测试方案，可以一次对几个不同的信号进行测试，但一般来说不可能同时测试到所有信号。试着设计一系列方案对所有多选器输出的上表中的缺陷进行测试(五个多选器输出的每一位都要测试到)。尽量使用较少的测试方案。

(1) 检查该信号上是否有固定为 0 缺陷

- a. `addi $t1 $zero 128` 如固定为 0, 则 \$t1 寄存器值为 0, 否则为 128
- b. `lw $t1 0($zero)` 如固定为 0, \$t1 寄存器值为 0, 否则为预设地址为 0 位置的值

(2) 能否只设计一个测试方案同时检查固定为 0 缺陷和固定为 1 缺陷

- a. 不能同时检查, 因为每条固定指令中第 7 位的值已经固定, 如果已经设置为 1, 则不能检查出固定为 1 缺陷

检测固定为 1 缺陷的指令 `addi $t1 $zero 0` 如固定为 1, 则 \$t1 寄存器值为 128, 否则为 0

- b. 不能检查出固定为 1 缺陷, 因为所有使得 `MemtoReg` 信号为 0 的涉及写寄存器的指令, `MemRead` 均为 0, 因此不能检验对应地址的数据是否被错误读出

(3) 将原来能在正常 MIPS 处理器上运行的程序做一些变换, 使之可以在这个处理器上运行

- a. 将使用立即数的指令中, 第 7 位为 0 的指令进行变换, 使得 `rs` 的值先减去 128, 再让立即数加上 128, 这样使得地址不变且指令中第 7 位恒为 1
- b. 不能变换, 因为 `MemtoReg` 为 1 将影响所有 R 型指令的写寄存器操作, 且没有相应的写寄存器的值可以替代

(4) 条件同第①问, 这次检测控制信号 `MemRead` 是否存在上表中的缺陷

- a. `MemRead` 固定为 1 对结果无影响, 即使出现此缺陷处理器仍然正常工作, 不能按前述方法检测
- b. 当 31-26 位均为 0 时, 此时为 R 型指令, 不涉及数据存储器读写, 因此出现此缺陷处理器仍正常工作, 不能按前述方法检测

(5) 条件同第①问, 这次检测控制信号 `Jump` 是否存在上表中的缺陷

- a. 将第一条指令设置为 `add $1 $0 $0`, 执行该指令后如果 PC 值为 0x00000004 则无该缺陷, 若为 0x00002048 则 `Jump` 存在该缺陷
- b. 涉及 `Jump` 信号的指令 31-26 位不均为 0, 因此不能检测该缺陷

(6) 设计一系列方案对所有多选器输出的上表中的缺陷进行测试

- a. `RegDst lw $1 0x1000($0)` 如内存值写入 \$1 则正常, 如写入 \$2 则证明存在缺陷固定为 1
`Jump add $1 $0 $0` 同(5)a
`Branch add $1 $0 $0` 若 PC 值为 0x00000004 则正常, 为 0x00002048 则存在缺陷固定为 1
`MemtoReg` 不能通过前述方法检查
`ALUSrc add $1 $0 $0` 若 \$1 为 0 则正常, 为 0x00002048 则存在缺陷固定为 1
- b. `RegDst add $2 $1 $1` 预设 \$1 为 1, 如 \$2 为 2 且 \$1 不变则正常, 如 \$2 不变且 \$1 为 2 则异常
`Jump` 不能通过前述方法检查
`Branch` 不能通过前述方法检查
`MemtoReg` 不能通过前述方法检查
`ALUSrc` 不能通过前述方法检查

6、本题讨论特定指令在单周期数据通路中的操作。根据下表中的 MIPS 指令分别回答下列问题(参考 MIPS 汇编作业中的附录)。

	指令
a.	<code>lw \$1,40(\$6)</code>
b.	<code>Label: bne \$1,\$2,Label</code>

(1)该指令字的值是多少?

(2)提供给寄存器堆“Read register 1”端口的寄存器号是多少? 该寄存器真的被读了吗?

对于“Read register 2”呢？

(3) 提供给寄存器堆“Write register”端口的寄存器号是多少？该寄存器真的被写了吗？

不同的指令需要设置数据通路上不同的控制信号。根据下表的两种控制信号情况分别回答下列问题(参考图 4)。

	控制信号1	控制信号2
a.	RegDst	MemRead
b.	RegWrite	MemRead

(4) 对于前面的指令而言，这两个控制信号的值应该是多少？

	指令
a.	lw \$1,40(\$6)
b.	Label: bne \$1,\$2,Label

(5) 对图 4 中的数据通路而言，写出控制单元中实现这两个信号的逻辑表达式。假设我们仅需支持 lw、sw、beq、add 和 j(jump)指令。

(1) 该指令字的值是多少

a. 100011 00110 00001 0000000000101000 即 0x8cc10028

b. 000101 00001 00010 1111111111111111 即 0x1422ffff

(2) 提供给寄存器堆“Read register 1”端口的寄存器号是多少 该寄存器真的被读了吗

a. 00110 即\$6 寄存器被读了

b. 00001 即\$1 寄存器被读了

对于“Read register 2”

a. 00001 即\$1 寄存器被读了

b. 00010 即\$2 寄存器被读了

(3) 提供给寄存器堆“Write register”端口的寄存器号是多少 该寄存器真的被写了吗

a. 00001 即\$1 寄存器被写了

b. 00010 即\$2 寄存器没有被写

(4) 对于前面的指令而言，这两个控制信号的值应该是多少

a. lw \$1 40(\$6) RegDst=0 MemRead=1

b. Label: bne \$1 \$2 Label RegWrite=0 MemRead=0

(5) 写出控制单元中实现这两个信号的逻辑表达式

a. $\text{RegDst} = \text{!OP5!OP4!OP3!OP2!OP1!OP0}$

$\text{MemRead} = \text{OP5!OP4!OP3!OP2OP1OP0}$

b. $\text{RegWrite} = \text{OP5!OP4!OP3!OP2OP1OP0} + \text{!OP5!OP4!OP3!OP2!OP1!OP0}$

$\text{MemRead} = \text{OP5!OP4!OP3!OP2OP1OP0}$

7、本习题讨论流水线对处理器时钟周期的影响。表中给出了数据通路中不同阶段延迟的两种情况，试根据这两种情况分别回答下列问题。

	IF	ID	EX	MEM	WB
a.	300ps	400ps	350ps	500ps	100ps
b.	200ps	150ps	120ps	190ps	140ps

- (1)流水线处理器与非流水线处理器的时钟周期分别是什么？
 - (2)lw 指令在流水线处理器和非流水线处理器中的总延迟分别是多少？
 - (3)如果可以将原流水线数据通路的一级划分为两级，每级的延迟是原级的一半，会选择哪一级进行划分？划分后处理器的时钟周期为多少？
- 假设处理器执行的指令比例下表两种情况所示，试根据每种情况分别回答下列问题。

	ALU	beq	lw	sw
a.	50%	25%	15%	10%
b.	30%	25%	30%	15%

- (4)假设没有阻塞和冒险，数据存储器的利用率是多少（占总周期数的百分比）？
- (5)假设没有阻塞和冒险，寄存器堆的写寄存器端口的利用率是多少？

(1) 流水线处理器与非流水线处理器的时钟周期分别是什么

- a. 流水线处理器：500ps 非流水线处理器：300+400+350+500+100=1650ps
- b. 流水线处理器：200ps 非流水线处理器：200+150+120+190+140=800ps

(2) lw 指令在流水线处理器和非流水线处理器中的总延迟分别是多少

- a. 流水线处理器：500×5=2500ps 非流水线处理器：1650ps
- b. 流水线处理器：200×5=1000ps 非流水线处理器：800ps

(3) 选择哪一级进行划分 划分后处理器的时钟周期为多少

- a. 选择 MEM 划分后的时钟周期为 400ps
- b. 选择 IF 划分后的时钟周期为 190ps

(4) 假设没有阻塞和冒险，数据存储器的利用率是多少（占总周期数的百分比）

- a. $15\%(lw) + 10\%(sw) = 25\%$
- b. $30\%(lw) + 15\%(sw) = 45\%$

(5) 假设没有阻塞和冒险，寄存器堆的写寄存器端口的利用率是多少

- a. $50\%(ALU)+15\%(lw) = 65\%$
- b. $30\%(ALU)+30\%(lw) = 60\%$

8、本习题讨论数据相关如何影响 4.5 节中基本五级流水线的运行。试根据下表的两种指令序列情况分别回答下列问题。

	指令序列		指令序列
a.	lw \$1, 40(\$6) add \$6, \$2, \$2 sw \$6, 50(\$1)	b.	lw \$5, -16(\$5) sw \$5, -16(\$5) add \$5, \$5, \$5

- (1)指出指令序列中存在的相关及其类型。
- (2)假设该流水线处理器没有转发，指出指令序列中存在的冒险并加入 **nop** 指令以消除冒险。
- (3)假设该流水线处理器中有充分的转发。指出指令序列中存在的冒险并加入 **nop** 指令以消除冒险。

根据下表的两种时钟周期情况，分别回答下列问题。

	无转发	充分的转发	仅ALU至ALU的转发
a.	300ps	400ps	360ps
b.	200ps	250ps	220ps

- (4)该指令序列在无转发和充分的转发时总执行时间分别是多少？后者相对于前者的加速比是多少。
- (5)如果仅有 ALU 至 ALU 的转发（没有从 MEM 到 EX 的转发），如何加入 **nop** 指令以消除可能的冒险？
- (6)该指令序列在仅有 ALU 至 ALU 的转发时总执行时间分别是多少？与无转发的情况相比，加速比是多少？

(1) 指出指令序列中存在的相关及其类型

- a. I1 和 I2: WAR, I1 和 I3: RAW, I2 和 I3: RAW
- b. I1 和 I2 I3: RAW, I1 I2 和 I3: WAR, I1 和 I3: WAW

(2) 假设该流水线处理器没有转发，指出指令序列中存在的冒险并加入 **nop 指令以消除冒险**

- a. I1 与 I3 \$1 数据更新需要间隔 2 个周期，I2 与 I3 \$6 数据更新需要间隔 2 个周期
在 add 后插入 2 个 **nop**
- b. I1 与 I2 \$5 数据更新需要间隔 2 个周期，I1 与 I3 \$5 数据更新需要间隔 2 个周期
在 lw 后插入 2 个 **nop**

(3) 假设该流水线处理器中有充分的转发，指出存在的冒险并加入 **nop 指令以消除冒险**

- a. 无需加入 **nop**
- b. I1 与 I2 数据更新需要间隔 1 个周期
在 lw 后插入 1 个 **nop**

(4) 该指令序列在无转发和充分的转发时总执行时间 后者相对于前者的加速比

- a. 无转发 $9 \times 300 = 2700$ 充分转发 $7 \times 400 = 2800$ 加速比 $= 2700 / 2800 = 0.964$
- b. 无转发 $9 \times 200 = 1800$ 充分转发 $8 \times 250 = 2000$ 加速比 $= 1800 / 2000 = 0.9$

(5) 如果仅有 ALU 至 ALU 的转发，如何加入 **nop 指令以消除可能的冒险**

- a. 在 add 后加入 1 个 **nop**
- b. 在 lw 后加入 2 个 **nop**

(6) 在仅有 ALU 至 ALU 的转发时总执行时间 与无转发的情况相比加速比

- a. 无转发 $9 \times 300 = 2700$ 充分转发 $8 \times 360 = 2880$ 加速比 $= 2700 / 2880 = 0.9375$
- b. 无转发 $9 \times 200 = 1800$ 充分转发 $9 \times 220 = 1980$ 加速比 $= 1800 / 1980 = 0.909$

9、本习题讨论指令集对流水线设计的影响。试根据下表的两条新指令回答下列问题。

a.	bezi (Rs), Label	if Mem[Rs] = 0 then PC = PC + Offs
b.	swi Rd, Rs(Rt)	Mem[Rs+Rt] = Rd

- (1) 为了将这条新指令增加到 MIPS 指令集，必须对流水线数据通路做什么改动？
- (2) 需要在(1)问的数据通路上增加哪些控制信号？
- (3) 对新指令的支持是否会引入新的冒险？已有冒险导致的阻塞是否会更加严重？

(1) 为了将这条新指令增加到 MIPS 指令集，必须对流水线数据通路做什么改动

a. 在 D 段设置比较模块，增加比较功能比较 Rs 的值和 0 是否相等，并输出比较结果；将指令立即数传入 NPC 模块，在 NPC 模块中增加加法器得出 PC+Offs 结果，并增加 Mux 的选择信号

b. 需将 Rd 数据传递到内存，在 E 段 ALU 后增加多选器选择 ALUresult 和 Rd 的值

(2) 需要在(1)问的数据通路上增加哪些控制信号

a. 增加判断与 0 是否相等的信号，当为 bezi 指令且比较结果相等时 PC 选择 PC+Offs

b. 在 E 段 ALU 后增加的多选器选择 1(Rd)

(3) 对新指令的支持是否会引入新的冒险

a. 会，bezi 新增控制冒险

b. 不会引入新的冒险

已有冒险导致的阻塞是否会更加严重

a. 如果加上到 D 段的转发以及开启延迟槽则不会，否则会使得已有阻塞更严重

b. 不会使已有阻塞更严重

10、

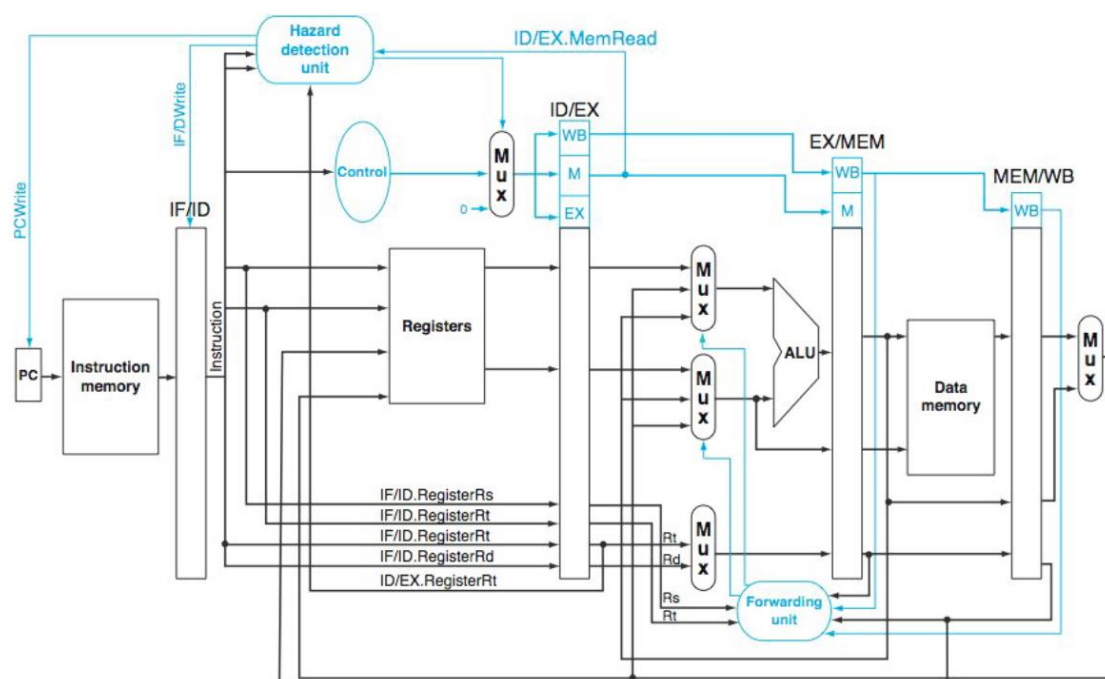


图 4

本习题讨论转发、冒险检测和指令集设计之间的关系。分别根据下表的两个指令序列回答下列问题。假设其在一个五级流水线上执行。

	指令序列		指令序列
a.	lw \$1, 40(\$6) add \$2, \$3, \$1 add \$1, \$6, \$4 sw \$2, 20(\$4) and \$1, \$1, \$4	b.	add \$1, \$5, \$3 sw \$1, 0(\$2) lw \$1, 4(\$2) add \$5, \$5, \$1 sw \$1, 0(\$2)

(1)如果没有转发或冒险检测电路，请插入 nop 指令以保证正确执行。

(2)重做(1)问，这次仅当通过改变或重排序指令都也不能避免冒险时才插入 nop 指令。假设可以使用寄存器 R7 作为临时寄存器。

(3)如果处理器中存在转发，但忘了实现冒险检测单元（以为实现了），代码执行时会发生什么情况？

(4)如果没有转发，对图 4 中的冒险检测单元来说还需要哪些新的输入输出信号？以该指令序列为例，说明为什么需要这些信号。

(1) 如果没有转发或冒险检测电路，请插入 nop 指令以保证正确执行

- a. 在 I1 后插入 2 个 nop，在 I3 后插入 1 个 nop
- b. 在 I1 后插入 2 个 nop，在 I3 后插入 2 个 nop

(2) 重做(1)问，这次仅当通过改变或重排序指令都也不能避免冒险时才插入 nop 指令

- a. I1: lw \$7, 40(\$6)
I3: add \$1, \$6, \$4
nop
I2: add \$2, \$3, \$7
I5: and \$1, \$1, \$4
nop
I4: sw \$2, 20(\$4)
- b. I1: add \$7, \$5, \$3
I3: lw \$1, 4(\$2)
nop
I2: sw \$7, 0(\$2)
I4: add \$5, \$5, \$1
I5: sw \$1, 0(\$2)

(3) 如果处理器中存在转发，但忘了实现冒险检测单元，代码执行时会发生什么情况

- a. I2 中\$1 的值并非 I1 中 lw 的值，从而程序执行错误
- b. I4 中\$1 的值并非 I3 中 lw 的值，从而程序执行错误

(4) 如果没有转发，对图 4 中的冒险检测单元来说还需要哪些新的输入输出信号

需要在 EX 中检查 R 型指令和 lw 指令的 Rd 寄存器，在 MEM 中检查目标寄存器号，因此需要添加 ID/EX 流水线寄存器的 Rd 和 EX/MEM 的输出寄存器作为输入信号

a. I1 中的\$1 为 EX/MEM 的输出寄存器，与 I2 中的\$1 产生冒险；I2 中的\$2 为 EX/MEM 的输出寄存器，与 I4 中的\$2 产生冒险

b. I1 中的\$1 为 ID/EX 的 Rd，与 I2 中\$1 产生冒险；I3 中的\$1 为 EX/MEM 的输出寄存器，与 I4 中的\$1 产生冒险

2021 秋《计算机硬件基础》

作业 07_高速缓存

1、某计算机的存储系统由 Cache 和主存组成。若所访问的字在 Cache 中，则存取它需要 10ns；将所访问的字从主存装入 Cache 需要 60ns。假定 Cache 的命中率为 0.9，计算该存储系统访问一个字的平均存取时间。

平均存取时间= $0.9 \times 10 + 0.1 \times (10 + 60) = 16\text{ns}$

2、假设一 4 路组相联 Cache，数据存储空间大小为 64KB，块大小为 16 字节，主存地址 32 位，主存一个字包含 4 个字节，Cache 采用写回策略，每个数据块包括 1 位有效位，Cache 每个字用 1 位脏位来表示是否被修改。

(1)CPU 如何解释主存地址（主存地址格式）(2)计算实现该 Cache 所需总存储容量

(1)主存地址格式

主存容量= 2^{32} 块大小= 2^4 Cache 组数= $2^{16}/2^4/2^2=2^{10}$ 组内块数= $2^{32}/2^{10}/2^4=2^{18}$

Tag(31-14)	Set(13-4)	Offset(3-0)
------------	-----------	-------------

(2)Cache 存储容量 = $(1(\text{有效位}) + 4(\text{脏位}) + 18(\text{Tag}) + 16 \times 8) \times 2^{12} = 75.5\text{KB}$

3、计算机系统包含 32K 字的主存，Cache 容量 4K 字，每组 4 Blocks，每 Block 64 个字。假设 Cache 开始是空的，CPU 顺序从存储单元 0，1，2 到 4351 中读取字，然后再重复这样的取数 9 次，Cache 速度是主存速度的 10 倍，采用 LRU 替换算法，假定块替换的时间忽略不计。

(1) 计算上述取数过程的命中率 (2)计算采用 Cache 后的加速比

主存容量= 2^{17} 块大小= 2^8 Cache 组数= 2^4 块内组数= 2^5 以二进制表示组号 0000-0011 组
每次循环装载 5 次 block，0100-1111 组只在第一次循环装载 4 次 block

(1)命中率 = $(12 \times 4 \times (10 \times 64 - 1) + 4 \times 5 \times 10 \times (64 - 1)) / 4352 \times 10 = 43272 / 43520 = 0.994$

(2)加速比 = $4352 \times 10 \times 10 / (43272 \times 1 + (43520 - 43272) \times 11) = 9.46$

4、考虑一个 Cache，其存取时间为 2.5ns，行大小为 64 字节，命中率 H=0.95。主存使用块传送方式，第一个字（4 字节）存取时间为 50ns，其后每个字存取时间为 5ns。

(1)出现一次 Cache 缺失的存取时间是多少？假设此时 Cache 等待，直到该行从主存传送到 Cache，然后再从 Cache 读取。

(2)假设行大小增大到 128 字节，命中率提升到 0.97，是否会降低平均存取时间。

(1) 出现一次 Cache 缺失的存取时间= $50 \times 1 + 5 \times 15 + 2.5 = 127.5\text{ns}$

(2) 初始平均存取时间= $0.95 \times 2.5 + 0.05 \times 127.5 = 8.75\text{ns}$

改进后平均存取时间= $0.97 \times 2.5 + 0.03 \times (50 \times 1 + 5 \times 31) = 8.65\text{ns}$

降低了平均存取时间

2021 秋《计算机硬件基础》

作业 08_虚拟存储器

1、给定一个 32 位的虚拟地址空间和一个 24 位的物理地址，对于下面不同的分页大小 P，请确定虚拟页号(VPN)、虚拟页内偏移量(VPO)、物理页号(PPN)和物理页内偏移量(PPO)的位数。

P	#VPN位数	#VPO位数	#PPN位数	#PPO位数
1KB	22	10		
2KB	21	11		
4KB	20	12		
8KB	19	13		

P	VPN	VPO	PPN	PPO
1KB	22	10	14	10
2KB	21	11	13	11
4KB	20	12	12	12
8KB	19	13	11	13

2、假定一个计算机系统有一个 TLB 和一个 L1 Data Cache。该系统按字节编址，虚拟地址 16 位，物理地址 12 位；页大小为 128 字节，TLB 采用 4 路组相联映射，共有 16 个页表项；L1 Data Cache 采用直接映射方式，块大小为 4 字节，共 16 行。在系统运行到某一时刻。TLB、页表和 L1 Data Cache 中的部分内容（用十六进制表示）如下图所示。

组号	标记	实页号	有效位	标记	实页号	有效位	标记	实页号	有效位	标记	实页号	有效位
0	03	—	0	09	1D	1	00	—	0	07	10	1
1	13	2D	1	02	—	0	04	—	0	0A	—	0
2	02	—	0	08	—	0	06	—	0	03	—	0
3	07	—	0	63	12	1	0A	34	1	72	—	0

(a) TLB 内容(4 路组相联，4 组，16 个页表项)

虚页号	实页号	有效位
000	08	1
001	03	1
002	14	1
003	02	1
004	—	0
005	16	1
006	—	0
007	07	1
008	13	1
009	17	1
00A	09	1
00B	—	0
00C	19	1
00D	—	0
00E	11	1
00F	0D	1

b) 部分页表内容（前 16 项）

行索引	标记	有效位	字节3	字节2	字节1	字节0
0	19	1	12	56	C9	AC
1	—	0	—	—	—	—
2	1B	1	03	45	12	CD
3	—	0	—	—	—	—
4	32	1	23	34	C2	2A
5	0D	1	46	67	23	3D
6	—	0	—	—	—	—
7	10	1	12	54	65	DC
8	24	1	23	62	12	3A
9	—	0	—	—	—	—
A	2D	1	43	62	23	C3
B	—	0	—	—	—	—
C	12	1	76	83	21	35
D	16	1	A3	F4	23	11
E	33	1	2D	4A	45	55
F	—	0	—	—	—	—

(C) L1 Data Cache 内容（直接映射，16 行，块大小 4 字节）

请回答下列问题：

- (1)虚拟地址中哪几位表示虚拟页号、哪几位表示页内偏移量？虚拟页号中哪几位表示 TLB 标记？哪几位表示 TLB 组索引？
- (2)物理地址中哪几位表示物理页号、哪几位表示页内偏移量？在访问 Cache 时，物理地址如何划分成标记字段、行索引字段和块内地址字段？
- (3)CPU 从地址 067AH 中取出的值是多少？要求对 CPU 读取地址 067AH 中内容的过程进行详细说明。

(1) $128=2^7$ 故页内偏移地址 7 位 虚页号 9 位

9 (虚页号)	7(页内偏移)
---------	---------

TLB 分为 4 组，故组索引位 2 位

7(TLB 标记)	2(组索引)	7(页内偏移)
-----------	--------	---------

(2) 物理地址与虚拟地址的页内偏移相等，故偏移 7 位，实页号 5 位

5 (实页号)	7(页内偏移)
---------	---------

块容量 $4=2^2$ Cache 大小= 2^6 区内索引 4 位 Tag 6 位

6(Tag)	4(Index)	2(offset)
--------	----------	-----------

(3) 地址 067AH = 0000011001111010B

虚页号：000001100 页内地址：1111010

首先在 TLB 中寻找实页号：TLB 标记为 0000011 组索引为 00，即组 0 的 03，有效位为 0

故在内存中寻找实页号：虚页号 000001100 即 00C 实页号为 19 即 11001

故物理地址为 11001111010 Tag:110011 Index:1110 即 E 行标记 33，在 Cache 中命中

offset 10 故取字节 2 中数据，所得数据为 4AH=01001010B

2021 秋《计算机硬件基础》

09 链接与运行 作业

1 (CSAPP 7.7) 不改变任何变量名字, 不改变原有赋值语句, 修改 bar5.c, 使得 foo5.c 输出 x 和 y 的正确值 (也就是整数 15213 和 15212 的十六进制表示)。

```
1  /* foo5.c */
2  #include <stdio.h>
3  void f(void);
4
5  int x = 15213;
6  int y = 15212;
7
8  int main()
9  {
10     f();
11     printf("x = 0x%x y = 0x%x \n",
12           x, y);
13     return 0;
14 }
```

```
1  /* bar5.c */
2  double x;
3
4  void f()
5  {
6     x = -0.0;
7 }
```

将 `double x`

改成 `static double x`

2 (CSAPP 7.9) 考虑下面的程序, 它由两个目标模块组成:

```
1  /* foo6.c */
2  void p2(void);
3
4  int main()
5  {
6     p2();
7     return 0;
8 }
```

```
1  /* bar6.c */
2  #include <stdio.h>
3
4  char main;
5
6  void p2()
7  {
8     printf("0x%x\n", main);
9 }
```

在 Linux 系统中编译和执行这个程序时, 即使函数 p2 不初始化变量 main, 它也能打印字符串 "0x55\n" 并正常终止。请解释原因。

在 foo6.c 中 main 为已定义的主函数, 属于强符号

在 bar6.c 中 main 为未初始化的变量, 属于弱符号

在链接时将所有的 main 都解释为强符号, 因此在函数 p2 中输出的 main 的值与 p2 是否初始

化无关

3 (CSAPP 7.10) a 和 b 表示当前路径中的目标模块或静态库, 而 $a \rightarrow b$ 表示 a 依赖于 b , 也就是说 a 引用了一个 b 定义的符号。对于下面的每个场景, 给出使得静态链接器能够解析所有符号引用的最小的命令行 (即含有最少数量的目标文件和库参数的命令)。

A. $p.o \rightarrow \text{libx.a} \rightarrow p.o$

B. $p.o \rightarrow \text{libx.a} \rightarrow \text{liby.a}$ and $\text{liby.a} \rightarrow \text{libx.a}$

C. $p.o \rightarrow \text{libx.a} \rightarrow \text{liby.a} \rightarrow \text{libz.a}$ and $\text{liby.a} \rightarrow \text{libx.a} \rightarrow \text{libz.a}$

A. `gcc p.o libx.a`

B. `gcc p.o libx.a liby.a libx.a`

C. `gcc p.o libx.a liby.a libx.a libz.a`

4. 考虑某目标文件 `m.o` 中对函数 `swap` 的调用。

```
9: e8 fc ff ff ff          call a<main+0xa>      swap()
```

具有如下重定位条目:

```
r.offset = 0xa
```

```
r.symbol = swap
```

```
r.type = R_386_PC32
```

A. 假设链接器将 `m.o` 中的 `.text` 重定位到地址 `0x80484e0`, 把 `swap` 重定位到地址 `0x80484f8`。那么 `call` 指令中对 `swap` 的重定位引用的值应该是什么?

B. 假设链接器将 `m.o` 中的 `.text` 重定位到地址 `0x80484d0`, 把 `swap` 重定位到地址 `0x8048500`。那么 `call` 指令中对 `swap` 的重定位引用的值应该是什么?

A. $\text{ADDR}(s)=\text{ADDR}(\text{.text})=0x4004e0$ $\text{ADDR}(r.\text{symbol})=\text{ADDR}(\text{swap})=0x4004f8$

$\text{refaddr}=\text{ADDR}(s)+r.\text{offset}=0x4004ea$

$*\text{refptr}=(\text{unsigned})(\text{ADDR}(r.\text{symbol})+*\text{refptr}-\text{refaddr})=0xa$

故重定位引用 `swap` 的值为 `0xa`

B. $\text{ADDR}(s)=\text{ADDR}(\text{.text})=0x4004d0$ $\text{ADDR}(r.\text{symbol})=\text{ADDR}(\text{swap})=0x400500$

$\text{refaddr}=\text{ADDR}(s)+r.\text{offset}=0x4004da$

$*\text{refptr}=(\text{unsigned})(\text{ADDR}(r.\text{symbol})+*\text{refptr}-\text{refaddr})=0x22$

故重定位引用 `swap` 的值为 `0x22`

5. 一位程序员编译运行下面两个文件：

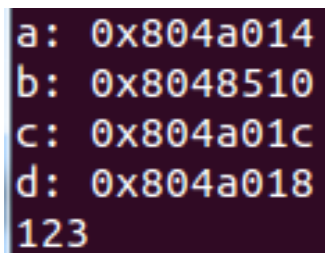
```
/* main.c */
#include <stdio.h>

long a = 1;
const long b = 2;
long c;
long d = -1;

int main(int argc, char *argv[]){
    printf("a: %p\nb: %p\nc: %p\nd: %p\n", &a, &b, &c, &d);
    printf("%ld\n", c);
    return 0;
}

/* data.c */
unsigned int c[2] = {123, 456};
```

看到了下面的输出：



```
a: 0x804a014
b: 0x8048510
c: 0x804a01c
d: 0x804a018
123
```

- a) 可执行文件的符号表中，由 main.c 生成的符号有几个？
 - b) main.c 中定义的强符号有哪些，弱符号有哪些？
 - c) b 的地址为什么离其它变量那么远？
 - d) 为什么 c 的地址在 d 后面？
- a) 由 main.c 生成的符号：a,b,c,d,main 共 5 个
- b) 强符号：a,b,d,main 弱符号：c
- c) b 为 const 常值，不放在.data 段而放在.rodata 段
- d) data.c 中定义了强符号 c 数组，因此打印地址时选择了强符号 c 而非 main.c 中定义的弱符号 c，故 c 的地址在 d 的后面

2021 秋《计算机硬件基础》

06 主存储器 作业

简答与设计：

1、(1) 说明存取时间与存取周期的区别。

(2) 什么是存储器的带宽？若某存储器的数据总线宽度为 64 位，存取周期为 100ns，则该存储器的带宽是多少？

(1) 存取时间：指读或写操作所用时间，即从给定地址到存储器完成读或写操作所需时间

存取周期：指两次访问存储单元间的最小时间间隔

(2) 存储器的带宽=随机访问存储器读写频率×字宽

读写频率 $f=1/T_C = 1/100\text{ns}=10^7\text{s}^{-1}$

带宽= $f \times 64=640\text{Mbps}$

2、某机字长 32 位，其存储容量是 64KB，按字编址其寻址范围是多少？若主存以字节编制，试画出主存字地址和字节地址的分配情况。

$64\text{KB}=2^{16}$ $32\text{bit}=2^5\text{B}$

按字编址 地址 14 位 寻址范围 000000000000000B-11111111111111B

也即 0x0000-0x3fff

按字节编址 地址 16 位

14(字地址)	2(offset)
---------	-----------

3、一个容量为 $16\text{K} \times 32$ 位的存储器，分别需要几条地址线 and 数据线？如果该存储器采用二维地址结构，且行地址和列地址的位数相同，则译码器输出的行选择线和列选择线分别有多少条？若选用下列不同规格的存储芯片来实现该存储器，需要各存储芯片的数目以及它们的排列方式分别是怎样的？

1K×4 位，2K×8 位，4K×4 位，16K×1 位，4K×8 位，8K×8 位

分别需要几条地址线和数据线？ $16\text{K}=2^{14}$ 地址线 14 条 数据线 32 条

译码器输出的行选择线和列选择线分别有多少条？

行地址列地址均 7 位，行选择线列选择线均 128 条

各存储芯片的数目以及它们的排列方式？

对于 $1K \times 4$ 位 需 128 个 其中每 8 个并联组成 32 位数据, 需这样 16 组构成 16K 地址
对于 $2K \times 8$ 位 需 32 个 其中每 4 个并联组成 32 位数据, 需这样 8 组构成 16K 地址
对于 $4K \times 4$ 位 需 32 个 其中每 8 个并联组成 32 位数据, 需这样 4 组构成 16K 地址
对于 $16K \times 1$ 位 需 32 个 其中每 32 个并联组成 32 位数据, 需这样 1 组构成 16K 地址
对于 $4K \times 8$ 位 需 16 个 其中每 4 个并联组成 32 位数据, 需这样 4 组构成 16K 地址
对于 $8K \times 8$ 位 需 8 个 其中每 4 个并联组成 32 位数据, 需这样 2 组构成 16K 地址

4、现有一容量为 $256K \times 8$ 的 DRAM 存储芯片, 试回答:

- (1) 该芯片包含多少个字单元?
- (2) 该芯片包含多少个二进制存储单元电路(存储位元)?
- (3) 该芯片的刷新地址计数器应该是多少位?
- (4) 若该 DRAM 芯片的存取周期为 $0.25\mu s$, 试问采用集中刷新、分散刷新及异步刷新三种方式的刷新间隔各为多少?

(1) $256K = 2^{18}$ 即 2^{18} 个字单元

(2) $256K \times 8 = 2^{21}$ 即 2^{21} 个存储位元

(3) 按行刷新, 故刷新地址计数器位数应与行地址位数相同, 即 9 位

(4) 集中刷新: 刷新间隔=刷新周期=2ms

分散刷新: 存取周期加长一倍 刷新间隔=存取周期 \times 刷新行数= $512 \times 0.25\mu s = 0.128ms$

异步刷新: 刷新间隔=刷新周期=2ms

5、设有一个 $64K \times 16$ 位的 RAM 芯片, 问该芯片共有多少个基本单元电路(简称存储基元)?
欲设计一种具有上述同样多存储基元的芯片, 要求对芯片字长的选择应满足地址线和数据线的总和为最小, 试确定这种芯片的地址线和数据线, 并说明有几种解答。

存储基元: $64K \times 16 = 2^{20}$ 个

对数据线个数讨论:

数据线 1 条, 地址线 20 条 总和 21

数据线 2 条, 地址线 19 条 总和 21

数据线 3 条, 地址线 19 条 总和 22

不妨设数据线个数为 x , 则地址线个数为 $(20 - \log_2 x)$ (向上取整)

总和为 $x - \log_2 x + 20$ 为单调递增函数, 因此可知随数据线个数增加, 总和增大

故数据线为 1 条或 2 条时有最小总和

即有两种解答:

①地址线 20 条 数据线 1 条

②地址线 19 条 数据线 2 条

6、设 CPU 有 16 根地址线，8 根数据线，并用 MREQ#（低电平有效）作访存控制信号，R/W#作读写命令信号（高电平为读，低电平为写），现有存储芯片 ROM ($2K \times 8$, $4K \times 4$, $8K \times 8$) 和 RAM($1K \times 4$, $2K \times 8$, $4K \times 8$)及 74138 译码器和其他门电路。试选择合适芯片，并画出 CPU 和芯片连接图。要求：

- (1) 最小 4K 地址为系统程序区，4096~16383 地址范围为用户程序区。
- (2) 指出选用的存储芯片类型及数量。
- (3) 画出片选逻辑。

(1) 系统程序区：0000H-0FFFH 用户程序区：1000H-3FFFH

(2) 系统程序区：2 个 $4K \times 4$ 并联 用户程序区：3 个 $4K \times 8$ 串联

(3) 片选逻辑：使用 74138 译码器 以 A12 A13 A14 为输入，通过 A12 A13 译码选择 Y0-Y3 信号作为片选信号，另外 A14 取非、A15、MREQ 信号分别接到 EN1、EN2A、EN2B，保证 A14、A15 均为 0 且 MREQ 为低电平时为有效选择

示意图如下：A0-A15 为地址总线 D0-D7 为数据总线

